




METHOD FOR MANAGING FILE

Patent number: JP2002082825
Publication date: 2002-03-22
Inventor: IWANO HIROTOSHI; IKEDA NATSUKO; NISHIMURA MOTOHIDE; KIYAMA JIRO; YAMAMURA HIROYUKI; YAMAGUCHI TAKAYOSHI; KITSUKE EIJI
Applicant: SHARP KK
Classification:
- international: G06F3/06; G11B20/12; G11B27/32; G06F11/14; G06F3/06; G11B20/12; G11B27/32; G06F11/14; (IPC1-7): G06F12/00; G06F12/16
- european: G06F3/06M; G06F17/30F; G11B20/12; G11B20/12D; G11B27/32D2
Application number: JP20010170443 20010606
Priority number(s): JP20010170443 20010606; JP20000188671 20000623

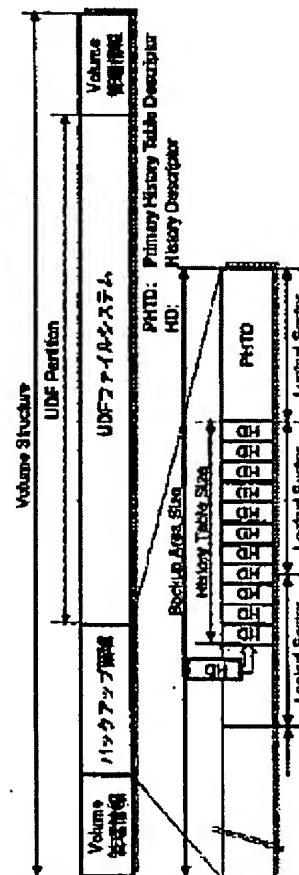
Also published as:

 EP1306761 (A1)
 WO0198905 (A1)
 US2003163449 (A1)

Report a data error here

Abstract of JP2002082825

PROBLEM TO BE SOLVED: To solve the problems that the number of maximum files which can be managed is limited due to the size of a management area because conversely, the management area and a management area for backup must preliminarily be secured in the case of preliminarily separating the management area from a real data area and performing backup by duplicating the management area in a file system and also that a method for performing duplication in each management can not be applied because there is no concept of a management area in such a file system where a management area and a data area are recorded in the same area.
SOLUTION: History information in which the file identifier of a file, the recording position of the file on a recording medium and the action type of the file are associated with one another is sequentially prepared and recorded on a history table area different from the recording area of the file and management information in the order of history information preparation, each time the actions of addition, change and deletion of the file to/in/from the recording medium.



Data supplied from the **esp@cenet** database - Worldwide

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-82825

(P2002-82825A)

(43)公開日 平成14年3月22日(2002.3.22)

(51)Int.Cl. ⁷	識別記号	F I	テ-マ-ト*(参考)
G 0 6 F 12/00	5 3 1	G 0 6 F 12/00	5 3 1 R 5 B 0 1 8
12/16	3 3 0	12/16	3 3 0 D 5 B 0 8 2

審査請求 未請求 請求項の数14 O L (全 32 頁)

(21)出願番号 特願2001-170443(P2001-170443)

(22)出願日 平成13年6月6日(2001.6.6)

(31)優先権主張番号 特願2000-188671(P2000-188671)

(32)優先日 平成12年6月23日(2000.6.23)

(33)優先権主張国 日本(J P)

(71)出願人 000005049
シャープ株式会社
大阪府大阪市阿倍野区長池町22番22号

(72)発明者 岩野 裕利
大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

(72)発明者 池田 奈津子
大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

(74)代理人 100102277
弁理士 佐々木 晴康 (外2名)

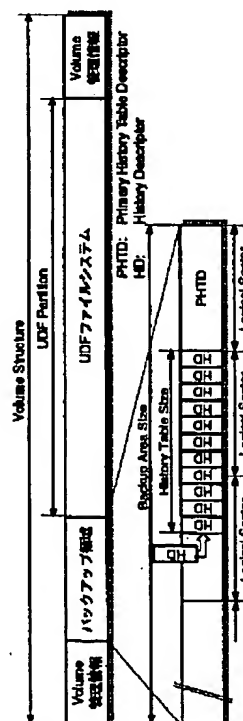
最終頁に続く

(54)【発明の名称】 ファイル管理方法

(57)【要約】

【課題】 ファイルシステムにおいて、管理領域と実データ領域を予め分離しておき、管理領域を2重化することでバックアップを行う場合、逆に管理領域及びバックアップ用の管理領域を予め確保しておく必要があるため、管理領域の大きさから管理できる最大ファイル数の制限がでてきてしまう。また、管理領域とデータ領域が同じ領域に記録されていくようなファイルシステムにおいて、管理領域という概念はないので、管理領域ごと2重化するという手法は適用できない。

【解決手段】 前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、当該ファイルのファイル識別子と当該ファイルの記録媒体上の記録位置情報及びファイルのアクション種別を対応づけた履歴情報を順次作成し、当該履歴情報を前記ファイル及び管理情報の記録領域とは異なる履歴テーブル領域に履歴情報の作成順に記録することで課題を解決する。



【特許請求の範囲】

【請求項1】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、当該ファイルのファイル識別情報と当該ファイルの記録媒体上の記録位置情報及びファイルのアクション種別を対応づけた履歴情報を順次作成し、当該履歴情報を履歴テーブル領域に履歴情報の作成順に記録することを特徴とするファイル管理方法。

【請求項2】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記履歴情報を前記ファイル及び管理情報の記録領域とは異なる履歴テーブル領域に履歴情報の作成順に記録することを特徴とする請求項1に記載のファイル管理方法。

【請求項3】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、ファイルとして管理される、前記履歴情報を記録する履歴テーブル領域に履歴情報の作成順に記録することを特徴とする請求項1に記載のファイル管理方法。

【請求項4】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記ファイル識別情報として、当該ファイルのファイル名あるいはファイルIDを含むことを特徴とする請求項1乃至請求項3に記載のファイル管理方法。

【請求項5】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記ファイル識別情報として、当該ファイルを管理する管理情報の記録媒体上の記録位置情報を含むことを特徴とする請求項1乃至請求項3に記載のファイル管理方法。

【請求項6】 入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、

前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記履歴情報として、当該ファイルを管理する管理情報の記録媒体上の記録位置情報を含むことを特徴とする請求項4に記載のファイル管理方法。

【請求項7】 前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴情報を記録装置のメモリ上に順次作成し、前記記録装置において、前記記録媒体の取り出し指示が行われた場合、当該履歴情報を該記録媒体上の履歴テーブル領域に記録することを特徴とする前記請求項1乃至請求項6に記載のファイル管理方法。

【請求項8】 前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴情報を記録装置のメモリ上に順次作成し、前記記録装置において、メモリへの電源供給切断指示が行われた場合、当該履歴情報を該記録媒体上の履歴テーブル領域に記録することを特徴とする前記請求項1乃至請求項6に記載のファイル管理方法。

【請求項9】 履歴テーブルの履歴確認指示に基づいて、前記履歴テーブル領域に記録された履歴情報のうち、所定のファイル識別情報を有する履歴情報のみを抽出することを特徴とする前記請求項1乃至請求項8のいずれかに記載のファイル管理方法。

【請求項10】 履歴テーブルの再構築指示に基づいて、前記履歴テーブル領域に記録された履歴情報のうち、同一のファイル識別情報を有する履歴情報を、当該履歴情報のアクション種別に応じて一つの履歴情報に統合し、履歴テーブル領域を縮小することを特徴とする前記請求項1乃至請求項9のいずれかに記載のファイル管理方法。

【請求項11】 前記管理情報は、ディレクトリの管理情報を含み、前記記録媒体へのディレクトリ情報の追加、削除のアクションが行われる毎に、当該ディレクトリの識別情報とアクション種別を対応づけた履歴情報を順次作成し、当該履歴情報を履歴テーブル領域に順次追記記録することを特徴とする前記請求項1乃至請求項10のいずれかに記載のファイル管理方法。

【請求項12】 前記履歴情報を履歴テーブル領域に順次追記記録する際、前記履歴テーブル領域の後方から前方方向へ順番に記録することを特徴とする前記請求項1乃至請求項11のいずれかに記載のファイル管理方法。

【請求項13】 ファイルの読出し指示に基づいて、前記管理情報を検索して、ファイルの読出しを行う際、当該ファイルの管理情報が読み出せない場合、前記履歴テーブル領域から、当該ファイルのファイル識別情報を有する最新の履歴情報を読み出し、当該履歴情報に基づいてファイルの読出しを行うことを特徴とする前記請求項1乃至請求項12に記載のファイル管理方法。

【請求項14】 管理情報の再構築指示に基づいて、前

記履歴テーブル領域から履歴情報を読みだし、当該履歴情報に基づいて、管理情報を生成し、当該管理情報を記録媒体に記録することを特徴とする前記請求項1乃至請求項13のいずれかに記載のファイル管理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、記録媒体におけるファイルのバックアップ等に関するファイル管理方法に関するものである。

【0002】

【従来の技術】PC用途やAV用途など様々な用途でディスク媒体にデータを記録する場合、論理ファイルシステムを用いる事が一般的である。論理ファイルシステムを用いる事によって、記録したデータをファイルとして管理し、ディレクトリ階層を構築する事が可能となり管理が容易となる。論理ファイルシステムとしては、広く普及しているFAT方式やDVDなどで導入されているUDF(Universal Disk Format)などが挙げられる。

【0003】論理ファイルシステムとは、ディスクに記録されたデータに対してデータを識別するための情報と、データが記録されたディスク上の位置情報などを管理情報としてディスクに記録し、それらの情報にアクセスする事によってファイルアクセスを可能とする仕組みの事である。例えば、論理ファイルシステムの管理情報には、ファイル名、そのファイルに関する作成日時、ファイルサイズ、状態を示す情報などの属性情報、対応する実データが記録されているディスク上の位置情報などが記録されている。FATやUDFなどと言った様々な種類の論理ファイルシステムが存在するが、これらの管理情報の構成や管理する属性情報が異なるだけで、ファイル名あるいはそれに準ずる情報からディスク上の目的のデータにアクセスができると言った意味では同じ目的のためのものである。

【0004】このような論理ファイルシステムを用いたディスクを通常使用している場合には全く問題は無いが、場合によってディスクに記録した情報が読み出せないと言った事が起こる可能性がある。例えば、何らかのショックによってディスク自体に傷が付いてしまったり、書き込み中にショックが加わり本来書き込むべき箇所でない所に書き込みを行ない記録内容を書き変えてしまったり、リムーバブルディスクの場合はディスクの表面に汚れが付着してしまったりする事によって発生する事が考えられる。

【0005】ディスクに書き込む前に問題のある箇所が発見できた場合には、ディスクの問題の箇所の代わりに代替領域に記録するディフェクトマネジメント機能を利用する事によって問題を回避できるが、情報を書いた後に問題が発生すると、その情報をディスクから読み出せなくなってしまう問題がある。記録したデータ自体が読

みだせない事も問題であるが、前述した論理ファイルシステムの管理情報がディスクから読みだせない事の方が問題となる。仮にあるファイルにアクセスするための論理ファイルシステムの管理情報が何らかの理由によって読み出せないと、例えばディスク上のデータに影響が無くても、対応するデータがディスクのどこに記録されていたかが不明となるため、データにアクセスできなくなってしまう。

【0006】PC用途で使用しているディスクでは、このような事が起こる可能性は稀である。これは、PC用途のドライブが物理的に安定した環境で使用されていることが多いためである。一方、AV用途でリムーバブルディスクを記録媒体としたビデオカメラなどを考えた場合、必ずしも物理的に安定した環境で使用されるとは限らない。ビデオカメラなので手に持って撮影するが、走りながらの撮影や、何かにぶつかったりとディスクにデータを記録している最中にショックが加わる事も考えられる。このように、PC用途で使用する場合と比較して苛酷な状況で使用される事が考えられ、前述したような予期せぬディスクからデータが読み出せないと言った状況が発生する確率が高くなる。

【0007】このような、論理ファイルシステムの管理情報が読み出せない事によって、データにアクセス出来なくなってしまう問題を解決するためには、論理ファイルシステムの管理情報を多重する事が考えられる。つまり、ディスク上に論理ファイルシステムの管理情報を多重する事によって、万が一ある管理情報が読み出せなくなった場合でも多重化してあるバックアップの管理情報を元に、データへのアクセスを可能とする事ができる。

【0008】図35にその様子を示す。このファイルシステムでは、ファイルシステムの管理情報とデータを記録する領域を区別し、それぞれがディスク上の対応する領域に記録される。ファイルシステムの管理情報が実際にはファイルが記録される領域とは別の所定の領域に記録されるので、管理領域は必ずこの領域内に記録されることになる。よって、この管理情報記録領域と全く同じ状態のバックアップ用の領域を用意する事によって管理情報を多重する事が可能となる。

【0009】

【発明が解決しようとする課題】上記した図35に示すように、管理領域と実データ領域を予め分離しておき、管理領域を2重化することでバックアップを行う場合、逆に管理領域及びバックアップ用の管理領域を予め確保しておく必要があるため、管理領域の大きさから管理できる最大ファイル数の制限ができてしまうという問題がある。

【0010】また、上記したFATやUDFファイルシステムは図34に示すように管理領域とデータ領域が同じ領域に記録されていくことになる。このようなファイルシステムにおいて、管理領域という概念はないので、

管理領域ごと2重化するという手法は適用できない。

【0011】また、ファイルシステムの管理情報はディレクトリ階層を構成するように管理されており、各管理情報の間にはこのディレクトリ階層に従った結び付き（ディスク上のアドレス位置でリンクされている）があるため、単純に管理情報を2重化することは容易ではない。例えば、ディレクトリを管理する管理情報にはそのディレクトリ内に定義されるファイルやディレクトリを管理する管理情報が記録されているアドレス位置が記述されている。よって、これらの論理ファイルシステムの管理情報を所定のバックアップ領域に単純にコピーするという方法で多重化する場合、管理情報間の連結情報であるディスク上のアドレスを、バックアップ領域内に記録する管理情報の記録位置に応じて変更しなければならない。

【0012】このような構成であると、ファイルの作成、変更、削除のタイミングで、実際の管理情報を更新するとともに、バックアップの管理情報のアドレスの更新処理を行う必要があり、バックアップ作成に多くの処理が必要となってしまふ。

【0013】本願は上記したような課題を解決するものであり、ファイルに関する作成、変更、削除、ディレクトリに関する作成、削除などといったイベントが発生した場合、そのファイル或いはディレクトリの識別情報と、当該ファイルのディスク上での位置あるいはディレクトリ情報を履歴情報として作成し、当該履歴情報を履歴テーブル領域に記録することで、管理情報が正常に読み出せない場合においても、容易にファイルを読み出すことを可能とし、また、バックアップ作成時に複雑な処理を行う必要がない。

【0014】

【課題を解決するための手段】本願の第1の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、当該ファイルのファイル識別情報と当該ファイルの記録媒体上の記録位置情報及びファイルのアクション種別を対応づけた履歴情報を順次作成し、当該履歴情報を履歴テーブル領域に履歴情報の作成順に記録することにより上記課題を解決する。

【0015】本願の第2の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記履歴情報を前記ファイル及び管理情報の記録領域とは異なる履歴テーブル領域に履歴情報の作成順に記録することにより上記課題を解決する。

【0016】本願の第3の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、ファイルとして管理される、前記履歴情報を記録する履歴テーブル領域に履歴情報の作成順に記録することにより上記課題を解決する。

【0017】本願の第4の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記ファイル識別情報として、当該ファイルのファイル名あるいはファイルIDを含むことにより上記課題を解決する。

【0018】本願の第5の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記ファイル識別情報として、当該ファイルを管理する管理情報の記録媒体上の記録位置情報を含むことにより上記課題を解決する。

【0019】本願の第6の発明によれば、入力されたデータをファイルとして記録媒体に記録し、各ファイルを少なくとも該ファイルの記録媒体上における記録位置情報を含む管理情報により管理する記録装置におけるファイル管理方法であって、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴テーブル領域に記録する前記履歴情報として、当該ファイルを管理する管理情報の記録媒体上の記録位置情報を含むことにより上記課題を解決する。

【0020】本願の第7の発明によれば、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴情報を記録装置のメモリ上に順次作成し、前記記録装置において、前記記録媒体の取り出し指示が行われた場合、当該履歴情報を該記録媒体上の履歴テーブル領域に記録することにより上記課題を解決する。

【0021】本願の第8の発明によれば、前記記録媒体へのファイルの追加、変更、削除のアクションが行われる毎に、前記履歴情報を記録装置のメモリ上に順次作成し、前記記録装置において、メモリへの電源供給切断指示が行われた場合、当該履歴情報を該記録媒体上の履歴テーブル領域に記録することにより上記課題を解決する。

【0022】本願の第9の発明によれば、履歴テーブルの履歴確認指示に基づいて、前記履歴テーブル領域に記

録された履歴情報のうち、所定のファイル識別情報を有する履歴情報のみを抽出することにより上記課題を解決する。

【0023】本願の第10の発明によれば、履歴テーブルの再構築指示に基づいて、前記履歴テーブル領域に記録された履歴情報のうち、同一のファイル識別情報を有する履歴情報を、当該履歴情報のアクション種別に応じて一つの履歴情報に統合し、履歴テーブル領域を縮小することにより上記課題を解決する。

【0024】本願の第11の発明によれば、前記管理情報は、ディレクトリの管理情報を含み、前記記録媒体へのディレクトリ情報の追加、削除のアクションが行われる毎に、当該ディレクトリの識別情報とアクション種別を対応づけた履歴情報を順次作成し、当該履歴情報を履歴テーブル領域に順次追記記録することにより上記課題を解決する。

【0025】本願の第12の発明によれば、前記履歴情報を履歴テーブル領域に順次追記記録する際、前記履歴テーブル領域の後方から前方方向へ順番に記録することにより上記課題を解決する。

【0026】本願の第13の発明によれば、ファイルの読出し指示に基づいて、前記管理情報を検索して、ファイルの読出しを行う際、当該ファイルの管理情報が読み出せない場合、前記履歴テーブル領域から、当該ファイルのファイル識別情報を有する最新の履歴情報を読み出し、当該履歴情報に基づいてファイルの読出しを行うことにより上記課題を解決する。

【0027】本願の第14の発明によれば、管理情報の再構築指示に基づいて、前記履歴テーブル領域から履歴情報を読みだし、当該履歴情報に基づいて、管理情報を生成し、当該管理情報を記録媒体に記録することにより上記課題を解決する。

【0028】

【発明の実施の形態】以下、本発明のディスク管理方法に関する実施形態について、図1乃至図34とともに詳細について説明する。本実施例は、ディスク装置として、AV記録再生を目的としたディスクを用いた携帯型のビデオカメラやビデオデッキ、PCに接続された外部記憶装置などを想定するものである。ディスク媒体は、リムーバブルディスクが好ましいが、ハードディスクなどの据え付け型であっても構わない。また、説明の都合上ディスクに用いる論理ファイルシステムとしてOSTA(Optical Storage Technology Association)の規格であるUDF(Universal Disk Format)を想定するが、その他の汎用論理ファイルシステムであっても構わない。

【0029】一般的なディスク装置の構成を図1に示す。データ入力出力部1はカメラなどから入力される映像信号を取り込んだり、再生するデータをモニタ等に出

力したりする。データ処理部2は、例えばMPEG符号をエンコードしたり、デコードしたりする信号処理等を行う処理部であり、処理されたデータはメモリ3に格納される。データを記録する場合は、ディスク制御部5において、ディスク6を制御しディスク上の目的の箇所にデータが記録され、再生時には、ディスク6を制御しディスク上の目的の箇所からデータが読み出されてメモリ3に格納される。各処理部はシステム制御部4によって制御される。

10 【0030】このようなディスク装置における論理ファイルシステムで、論理ファイルシステムの管理情報とデータを記録する領域が明確にわかれている場合、管理情報の領域を多重する事によって容易に管理情報のバックアップを持つ事が可能である。しかしながら、論理ファイルシステムの管理情報とデータを記録する領域が明確に分かれていない場合、つまり論理ファイルシステムの管理情報とデータが同一のディスク空間に記録される場合は、管理情報のバックアップを容易に持つ事はできない。

20 【0031】管理情報とデータを記録する領域が明確に分かれている場合とは異なり、ディスクに分散している管理情報のある特定の大きさのバックアップ用領域に記録する事ができないからである。これは、ディレクトリの管理情報にそのディレクトリに含まれるファイルやディレクトリの管理情報が記録されているディスク上のアドレスが記述されており、バックアップ領域に管理情報をバックアップする場合、論理ファイルシステムの管理情報間の連結情報であるディスク上のアドレスを、バックアップ領域内の管理情報の記録位置に応じて変更しなければならない事を意味する。

30 【0032】そこで、本発明では特にファイルやディレクトリなどの管理情報を多重して持つ機能の無い汎用の論理ファイルシステムにおいて、管理情報が読み出せなくなる事によるデータの読み出しが不可能になる事を防ぐ事を目標としている。

40 【0033】既にディスクに記録されているファイルやディレクトリはUDFのような論理ファイルシステムによって管理されているので、論理ファイルシステムの管理情報のバックアップを取るのにあまり複雑な操作が発生してしまつては意味が無い。そこで本発明では、バックアップ情報として、ファイルやディレクトリにアクセスするのに必要最低限の情報のみをバックアップするものとする。この必要最低限のバックアップ情報をバックアップの対象となるファイルシステムの管理する領域とは別の領域を用意してその領域中に記録する。

50 【0034】図2に、本発明で用いるバックアップ領域とUDF規格によって規定されるパーティションの関係を示す。UDFでは論理セクタ番号256とディスク上の最後の論理セクタ番号-256にAnchor Volume Descriptor Pointerが記

録される事が決まっており、この情報にアクセスする事によって、Volume Descriptor Sequenceを把握する事が可能となる。論理セクタ番号とは、ユーザシステムがアクセスできるディスク上の空間に昇順に付加されたアドレスの事である。

【0035】また、Anchor Volume Descriptor Pointerには、Volume全体を管理する管理情報で構成される、Volume Descriptor Sequenceの記録位置が管理されている。Volume Descriptor Sequenceには、Volume内に定義されるパーティションに関する管理情報が管理されており、実際にファイルやディレクトリを作成するUDFのファイルシステムを構築するパーティションの位置情報を取得する事が可能となる。本発明における、履歴情報を記録するファイル及び管理情報の記録領域とは異なる領域とは、このUDFパーティション外に確保する領域の事を指す。

【0036】このバックアップ情報用領域の位置情報は、例えば、論理セクタ番号128に記録される本発明用のAnchor Descriptorによって把握する事が可能となる。また、バックアップ情報用領域の領域の位置を把握するためのAnchor Descriptorを用いなくて、ある特定の固定位置に領域があると決めて確保しても構わない。図の例ではVolumeの管理情報のバックアップであるReserved Volume Descriptor SequenceがMain Volume Descriptor Sequenceの後も記録されている。

【0037】また、図3にバックアップ領域の中の様子を示す。図中の先頭のVolume管理情報は、図2においてバックアップ情報用領域の手前までの領域に対応する。バックアップ領域には、Primary History Table Descriptor (PHTD)と複数のHistory Descriptor (HD)が記録されている。全てのHDを合わせてHistory Tableと呼ぶ事とする。このHistory TableがUDFファイルシステムの管理情報のバックアップ情報(履歴情報)となる。PHTDはバックアップ領域中の最後の論理セクタに記録され、HDはPHTDの1つ前の論理セクタからバックアップ領域の先頭方向に向かって順次追記されて行く。論理セクタ内では、後ろからHistory Descriptorをつめて記録することになる。

【0038】例えば、初めて52byteの大きさのHistory Descriptorを記録する場合、PHTDの1つ前の論理セクタ内で1996byte目(セクタサイズ2048byte-History Descriptorの大きさ52byte)からHistory Descriptorを記録することにな

る。このように記録するのは、論理セクタ番号の低い方から最新のHistory Descriptorが順番に記録されているため、バックアップ情報へのアクセスが容易になるためである。

【0039】例えば、任意のファイルのバックアップ情報であるHistory DescriptorをHistory Tableから抽出する場合、このように記録されていることによって、ディスクから読み出したHistory Tableを格納するメモリ上において、メモリ空間上で先頭から時間的に最新のHistory Descriptorが配置されることになり、アクセスするには都合が良い。目的のHistory Descriptorが比較的新しく追加になっていれば、すぐに見つかることになる。また使用できるメモリの制限から一度に全てのHistory Descriptorを読み出せないような場合にも非常に有効である。

【0040】UDFファイルシステムにおいてファイルが新規に作成されたり、変更が発生したり、削除されたり、またディレクトリの場合は新規に作成されたり、削除された場合、そのファイルやディレクトリ毎に1つのHistory DescriptorがHistory Tableに追加される。PHTDには、History Tableの大きさが管理されており、常にHistory Tableの最後尾(ディスク上では最前部)が分かるようになっているので、History Descriptorの追加は容易に行なう事が可能である。UDFなどの汎用ファイルシステムと異なり今までに記録されたHistory Descriptor(ファイルやディレクトリのバックアップ情報)をHistory Tableからは削除しない。

【0041】つまり、UDFファイルシステムにおいてファイルやディレクトリの作成、変更、削除などといったイベントが発生する度に、単純にそれらのイベントに対応するHistory DescriptorがHistory Tableの最後尾(ディスク上では最前部)に追加されるだけである。

【0042】仮に、UDFファイルシステムの管理情報が読み出せないと言った状況になった場合、目的のファイルやディレクトリを特定するための識別情報をキーとして、History Tableの最後尾(ディスク上では最前部)から順番に、対応するHistory Descriptorが見つかるまで見ていく。探し出したHistory Descriptorを参照する事によって、対応するファイルが記録されているディスク上の位置情報を把握する事ができ、データにアクセスできないと言った問題点を解決する。

【0043】目的のHistory Descriptorを探し出すのに、History Tableをサーチしなければならないが、本発明の目的であるバック

アップ用途であり、この情報にアクセスする場合は非常時である事を考慮すれば許容できるものである。その反面、UDFファイルシステムにおけるファイルやディレクトリの作成、変更、削除と言ったイベントが発生した際のHistory Tableの更新は、History Descriptorの追加のみと必要最低限のデータ量および処理に抑えられている事が特徴となる。

【0044】またHistory Tableには、History Tableを作り始めてからのUDFファイルシステムにおけるファイルの作成、変更、削除、またディレクトリの作成、削除と言ったイベントに対応するHistory Descriptorが記録されている事になるので、History Tableは単純なUDFファイルシステムの管理情報のバックアップだけではなく、ファイルシステムの管理情報の更新履歴としても利用する事が可能である。

【0045】History DescriptorはHistory Tableに追加されて行くだけなので、使用過程においてHistory Tableが巨大になってしまったり、バックアップ領域の空きが残量が無くなってしまう事も考えられる。そこで、History TableからUDFファイルシステムで定義されているファイルやディレクトリに対応するHistory Descriptorだけ残してその他の過去の更新履歴情報であるHistory Descriptorを全て削除しHistory Tableを再構築する事も可能である。

【0046】History Descriptorで管理するバックアップ対象のファイルやディレクトリを特定するための識別情報としてファイル名を用いる第1の実施の形態について説明を行なう。ここで、図4にPrimary History Table Descriptorの内容を示す。Primary History Table Descriptorは、History Tableを管理するための情報やバックアップ領域の情報を管理する記述子である。

【0047】Area Sizeはバックアップ領域の大きさをbyte数で表し、Uint32型として記録される。Last HD Added Timestampは、History DescriptorがHistory Tableに追加された最終日時を記録する。この情報によって、最後にHistory Descriptorを追加した日時を把握する事ができ、例えばUDFファイルシステムの管理情報との整合性を見る場合などに利用する事ができる。Last HT Updated Timestampは、History Tableを最後に再構築した日時を記録する。この情報によりHistory Tableが保持する更新履歴がいつからのものであるかを把握する事が可能となる。

【0048】Number of History Descriptorsは、History Tableに記録されているHistory Descriptorの数を示し、Uint32型で記録される。History Table Sizeは、History Tableのサイズをbyte数で表しUint32型で記録される。History Table中の全History Descriptorの合計サイズをByte数で表す。ディスクへのHistory Descriptorの追加は論理セクタ単位で行うのが一般的なので、仮に1論理セクタのサイズが2KBとすると、このHistory Table Sizeを2KBで割り、その商がアクセスすべきバックアップ領域の最後の論理セクタから1セクタ(PHTDのサイズ)引いた箇所からみた論理セクタ数となる。

【0049】また、余りが追加を行うべき論理セクタに既にデータが記録されている量である。つまり、追加を行う場合には目的の論理セクタを一度読み出して、既に記録されている情報の前にHistory Descriptorを追加して、その論理セクタの内容をディスクに記録する事になる。

【0050】なお、図中の、RBPはRelative Byte Positionを意味し、先頭から見た対応する管理項目の開始位置を示す情報で、Lenはその管理項目の大きさをByteで表し、Field Nameは管理項目名、Contentsは、管理項目がどのような形式で記録されなければならないかということを示す。Contentsで用いられているデータ型のうち、Uint8は符号無し8bit整数、Uint16は符号無し16bit整数、Uint32は符号無し32bit整数を意味する。Stringは文字列を格納するためのデータ型、Timestampは日時情報を格納する型である。

【0051】図5にHistory Descriptorの内容を示す。History Descriptorは、論理ファイルシステムにおけるファイルやディレクトリの管理情報のバックアップ情報であり、対応するファイルやディレクトリにアクセスするための必要最低限の情報で構成されている。File Sizeは、バックアップの対象となるファイルのファイルサイズをbyte数で示し、Uint32型で記録される。具体的には、対応するUDFファイルシステムのFile Entry内のファイルサイズの情報を示すInformation Lengthと同じ値を記録する。Modification Date and Timeは、このファイルが追加、変更、削除、またディレクトリが追加、削除された時刻を示し、Timestamp型で記録される。バックアップ対象がファイルの場合、対応するUDFファイルシステムのFile Entry内の変更日時を示すModification Date

and Timeと同じ値を、ディレクトリの場合はFile Entry内の作成日時を示すAccess Data and Timeと同じ値を記録する。また、ファイルやディレクトリの削除を示すHistory Descriptorの場合は、実際の削除の日時を記録する。

【0052】LBN of FEは、バックアップ対象となるファイルやディレクトリのUDFファイルシステムでのFile Entryの記録位置を示すUDFパーティション内の論理ブロック番号を示し、Uint32型で記録する。Attributesは、バックアップのイベント種別およびバックアップの対象がファイルなのかディレクトリなのかをUint16型で示す。Bit0は、バックアップ対象がファイルなのかディレクトリなのかを示し、0の場合はファイル、1の場合はディレクトリを示す。Bit1と2は、符号無し2bitの情報として扱い、この2bitが0の場合「作成」、1の場合は「変更」、2の場合は「削除」を示す。

【0053】Length of File Identifierは、ファイルやディレクトリを識別するための情報であるFile Identifierの長さをbyte数で示し、Uint16型で記録される。Length of Allocation Descriptorsは、Allocation Descriptorsフィールドの長さをbyte数で示し、Uint32型で記録する。バックアップ対象であるファイルのUDFファイルシステムでのFile EntryのLength of Allocation Descriptorsと同じ値を記録する。

【0054】File Identifierは、ファイルやディレクトリを識別するための情報であり、Length of File Identifier byteだけstring形式で記録される。ファイルやディレクトリを識別するための情報とは、通常ファイル名やディレクトリ名である。ここで、File Identifierにはファイルやディレクトリのパス名を併記して記録するものとする。

【0055】例えば、Rootディレクトリ下のDATAディレクトリ下のFILE1.DATというファイルの場合は、¥DATA¥FILE1.DATと記録する。また、前述のLength of File Identifierはこの場合、15byteとなる。File Identifierはファイル名である必要はなく、例えばファイルID番号などファイルやディレクトリが特定できるものであれば構わない。ただし、同一ディスクに色々な種類のFile Identifierが混在できるという意味ではない。

【0056】Paddingは、Allocation Descriptorsフィールドの開始RBPが4

byteアライメントされるように調整を行う情報であり、必要な数だけは00hを記録する。 $4 \times ip(L_FI + 28 + 3) / 4 - (L_FI + 28)$ によってPaddingすべきbyte数が求まる。ここで、 $ip(n)$ はnの整数部分を返す関数であり、L_FIはLength of File Identifierフィールドによって示される値である。Allocation Descriptors (AD)はディスク上のデータの記録位置を管理するための管理構造であり、Uint32型のExtent Lengthと、Uint32型のExtent Positionによって構成される。ここでは、Extent LengthとExtent Positionを合わせてShort_ad型として扱うものとする。

【0057】Extent Lengthは、分断の長さをbyte数で示し、Extent Positionは分断の開始論理ブロック番号が記録される。1つのファイルであってもファイルに対応する実データはディスク上で分断されて記録する事も可能であるため、ファイルを構成する分断の数だけAllocation Descriptorsが記録されることになる。

【0058】実際には、1つのAllocation Descriptorsが8byteなので、Length of Allocation Descriptorsで示される値を8で割ることによってAllocation Descriptorsの数が求まる。このフィールドには、バックアップ対象のUDFファイルシステムでのFile Entry内のAllocation Descriptorsと同じ内容を記録する。

【0059】以上のような管理情報を用いた実際の実施例を説明して行く。図6に示すようなディレクトリ階層のファイルやディレクトリがある場合、対応するUDFファイルシステムが定義するパーティション内の論理ファイルシステムの管理情報とデータ配置の様子の例を図7に示す。

【0060】パーティション内の先頭から、パーティション内の空き領域情報を管理するSpace Bitmap Descriptor (SBD)、ファイルシステムの基本管理情報であり、Rootディレクトリを管理するFile Entry (FE Root)へのポインタ情報を持つFile Set Descriptor (FSD)、File Set Descriptorが終った事を示すTerminating Descriptor (TD)が記録されている。

【0061】また、Rootディレクトリ、DATAディレクトリをそれぞれ管理するFile Entry (FE)と、各ディレクトリの下に定義されるファイルのファイル名や属性情報、そしてそのファイルあるいは下位のディレクトリを管理するFile Entry

(FE)へのポインタ情報であるFile Identifier Descriptor (FID)、Rootディレクトリの下に定義されているREADME.TXTを管理するFile Entry (FE)とDATAディレクトリの下に定義される、FILE1.DATとFILE2.DATを管理するFile Entry (FE)が記録されている。

【0062】なお、History Descriptor内のLBN of FEフィールドには、このファイルを管理するFile Entryが記録された論理ブロック番号が記録されることになる。ファイルおよびディレクトリのFile Entryはそれぞれ対応するデータが記録されているディスク上の分断情報をExtentとしてAllocation Descriptorsによって管理している。例えば、図7の例においては、FILE1.DATに対応するデータはディスクでは分断して記録されており、Extent4および5によって構成されている。またFILE2.DATに対応するディスク上のデータは連続的に記録されており、Extent6によって構成されている。

【0063】このような状況のディレクトリ階層に対応するHistory Tableの例を図8に示す。この図では、一番下にPrimary History Table Descriptorが配置されており、History Tableの最後尾は図の一番上に相当する。この例では、Rootディレクトリ(¥)、¥README.TXT、¥DATAディレクトリ、¥DATA¥FILE1.DAT、¥DATA¥FILE2.DATの順番で作成された様子を示している。一番下の枠を除いて1つの枠が1つのHistory Descriptorに対応し、全てのHistory DescriptorをまとめてHistory Tableを構成する。

【0064】図9に図8に示す状態から、DATAディレクトリの下にFILE3.DATというファイルが作成(追加)された場合の様子を示す。FILE3.DATがUDFファイルシステムで作成されると、それに対応するHistory DescriptorがHistory Tableの最後尾(ディスク上では最前部)に追加される。

【0065】この際、History DescriptorのFile Identifierには¥DATA¥FILE3.DATが記録され、Timestampにはファイルの作成された日時、Attributeにはファイルおよび作成を示す0000h、File SizeにはFILE3.DATのファイルサイズを、Length of Allocation Descriptorsには、FILE3.DATに対応するデータのディスク上での分断数を8倍した値を、そしてAllocation Descriptors (AD)

にはそれぞれの分断に関する位置情報を記録する事になる。ここでは、FILE3.DATはExtent7のみによって構成されている(連続して記録されている)ことになる。

【0066】図10に、図9で示された状態から、さらにDATAディレクトリの下でFILE2.DATの内容が変更された場合の様子を示す。UDFファイルシステムにおいてFILE2.DATのディスク上の位置が変更になったり、内容が変更になった場合、History Tableの最後尾(ディスク上では最前部)に、対応するHistory Descriptorを追加する。この際、Attributeにはファイルおよび変更を示す0002hを記録する。また、FILE2.DATのディスク上の位置が変更となり、Allocation Descriptors (AD)が管理する分断がExtent8に変更になる。

【0067】このFILE2.DATの変更に關するHistory Descriptorが追加になった時点で、History Table内に既に存在している、FILE2.DATを作成した際のHistory Descriptorが古い情報となり、更新履歴情報となる。

【0068】図11に、図10で示された状態から、さらにDATAディレクトリの下でFILE1.DATを削除した場合の様子を示す。UDFファイルシステムにおいてFILE1.DATが削除された場合、History Tableの最後尾(ディスク上では最前部)に対応するHistory Descriptorを追加する。この際、Attributeにはファイルおよび削除を示す0004hを記録し、Allocation Descriptors (AD)には、削除する直前のデータの位置情報を記録する。

【0069】削除されるデータであるため直前の記録位置を記録しないようにしても良い。このFILE1.DATの削除に関するHistory Descriptorが追加になった時点で、History Table内に既に存在しているFILE1.DATを作成した際のHistory Descriptorが古い情報となり、更新履歴情報となる。

【0070】なお、図9乃至11においてイベント種別が作成、変更、削除について説明したが、ファイルシステム上では例えば、ファイルやディレクトリのコピーや移動あるいは名称変更なども起きる。これらのイベントは基本的に前述の作成、変更、削除を組み合わせることによって更新情報を表現する事が可能である。

【0071】例えば、あるファイルの移動であれば、移動元のファイルに対応する削除を表すHistory Descriptorを作成し、移動先のファイルに対応する作成を表すHistory Descriptorを作成する。また、同様に名称変更でも、名称を変更

する前のファイルに対応する削除を表すHistory Descriptorを作成し、名称変更後のファイルに対応する作成を表すHistory Descriptorを追加すれば良いことになる。

【0072】ここで、History Tableを再構築する場合の様子を図12に示す。History Tableには、UDFファイルシステムにおいてファイルに関する作成、変更、削除、またディレクトリに関する作成、削除と言ったイベントが発生する度にHistory Descriptorが単純に追加になっていく。History Descriptorを記録する領域は決まっているので、状況に応じてHistory Tableから不要な情報を削除しHistory Tableを再構築する事も必要となる。

【0073】この場合は、History Tableの最後部（ディスク上では最前部）のHistory Descriptorから順番に見て行き、同一のファイルやディレクトリに関する情報つまり更新履歴に相当するHistory DescriptorをHistory Tableから削除する。また、Attributeが削除のHistory Descriptorに関しても同様に削除する。図の例では、左側のHistory Tableから更新履歴に相当する「¥DATA¥FILE2. DAT作成」と「¥DATA¥FILE1. DAT作成」と、「¥DATA¥FILE1. DAT削除」のHistory Descriptorを削除し、右のHistory Tableのように再構築する。この際、残ったHistory DescriptorのAttributeは全て作成に変更する事になる。

【0074】ここで、処理の詳細をフローチャートで説明する。図13にUDFファイルシステムにおいてファイルやディレクトリの作成、変更、削除と言ったイベントが発生した場合の処理の流れを示す。

【0075】ステップS1において、ファイルの作成、変更、削除イベントが発生すると、ステップS2においてPrimary History Table Descriptorを読み出し、History Tableの最後部（ディスク上では最前部）を把握する。

【0076】具体的には、History Table中のHistory TableSizeからHistory Tableの最後尾の位置を把握する事が可能となる。ステップS3において、ファイルに対応するHistory DescriptorをHistory Tableの最後に追加する。

【0077】この際History Descriptorには、図5に示すように、ファイルの名前をフルパス形式でFile Identifierに、発生したイベントの種類およびファイルかディレクトリなのかをAttribute、イベントが発生した日時、イベン

トが発生した対象のUDFファイルシステムにおけるファイルの管理するディスク上の分断を管理する形で、AllocationDescriptorsが記録される。ステップS4において、PrimaryHistory Table Descriptor内の最後尾（ディスク上では最前部）にHistory Descriptorを追加した時刻を示すLast HD Added Timestampと、History Descriptor数、History Table Sizeを更新して処理を終了する。UDFにおけるファイルに関するイベントが発生した際の処理は以上のように、単純なHistory Descriptorの追加処理となる。

【0078】ファイルの場合とは異なり、バックアップ対象がディレクトリの場合は、ディレクトリの存在のみをHistory Descriptorで管理することにし、File Size、LBN of FEやAllocationDescriptorsは記録しないようにしてもよい。

【0079】このフローチャートでは、UDFのファイルシステムにおいてファイルの作成、変更、削除、またディレクトリの作成、削除と言ったイベントが発生する度に対応するHistory DescriptorをHistory Tableに追加する説明になっているが、例えば電源投入時やディスクを装着した時にPrimary History Table Descriptorを読み出し、電源を切断する時やディスクを排出する時などにその期間中に発生した全てのイベントに対応するHistory DescriptorをまとめてHistory Tableに追加しPrimary History Table Descriptorを更新し記録しても良い。つまりHistory Descriptorをメモリ上に保持しておき、あるタイミングで一度にディスク上のHistory Tableを更新しても構わないものとする。

【0080】このように、ファイルやディレクトリに関するイベントが発生する度にHistory Descriptorを追記する訳である。UDFにおいてファイルやディレクトリにアクセスする場合、まずFile Identifier Descriptorにアクセスしディレクトリ内に記録されているファイルやディレクトリ名を把握する。アクセスしたいファイルやディレクトリに対応するFile Identifier Descriptorの情報より、対応するFile Entryの記録位置を把握し、File Entryにアクセスする。

【0081】File Entryにはファイルやディレクトリの実体の記録位置が管理されているので、この情報を元に実体にアクセスする訳である。このときファイルやディレクトリにアクセスしようとしたが、それら

を管理するFile Entryにアクセスできない非常時に、バックアップ情報であるHistory Tableにアクセスする手段に関して図14のフローチャートを用いて説明する。

【0082】ステップS10において、バックアップ情報であるHistory Tableへのアクセス要求が発生すると、ステップS11において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置（ディスク上では最前部）を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展開される。ステップS12において、History Table中の最後（時間的には最新）のHistory Descriptorに注目をする。

【0083】ステップS13において、今アクセスしたいファイルあるいはディレクトリのフルパス形式の名前と注目しているHistory DescriptorのFile Identifierが一致するかどうかを判定する。一致しない場合はステップS14において全てのHistory Descriptorをサーチしたかどうかを判定する。全てのHistory Descriptorをサーチした場合は、探そうとしているバックアップ情報が見つからないという事になり、ステップS17においてエラー処理を行ない処理を終了する。ステップS14においてまだ全てのHistory Descriptorをサーチし終わっていない場合は、ステップS15において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS13に戻り処理を繰り返す。

【0084】ステップS13において名前が一致すると判定された場合は、ステップS16において注目しているHistory Descriptorのイベント種別を示すAttributeが削除であるかどうかを判定する。イベント種別が削除の場合、探そうとしているファイルあるいはディレクトリのバックアップ情報が削除状態を示すものであるためステップS17においてエラー処理して処理を終了する。

【0085】ただし、ここで、ファイルやディレクトリの削除した時点での情報が取得したいのであれば、エラー処理を行わず抽出したHistory Descri

ptorを利用しても構わない。ステップS16においてイベント種別が削除以外であれば、注目しているHistory Descriptorが探し出すべきバックアップ情報であり、サーチ処理を終了する。探し出したHistory Descriptor中のAllocation Descriptorsの情報をを用いてファイルやディレクトリのデータにアクセスが可能となるわけである。

【0086】このように、ある特定のファイルやディレクトリのUDFファイルシステム管理情報にアクセスできない場合に、History Tableから対応する最新のHistory Descriptorを抽出し、その情報から対応するデータの記録位置を把握しデータにアクセスする事を可能とした。また、この情報を利用して主の管理情報であるUDFの管理情報を再構築する事も可能である。例えば、あるファイルを管理するUDFのFile Entryが読み出せなくなった場合は、対応するバックアップ情報に相当するHistory Descriptorの情報を元にFile EntryをUDFのパーティション内に記録し直す。

【0087】File Entryの情報としては、作成時刻はFile Entryを作成し直した時刻、属性情報は標準的な情報にセットされ、問題の発生する前の状態とは必ずしも同じでは無いが、データにアクセスするための必要最低限の情報であるデータの記録位置情報に関しては完全に復活できるわけである。復旧したUDFのFile Entryの記録位置が問題のあった元のFile Entryの記録位置と異なる場合、File Entryの記録位置を管理しているUDF管理情報のFile Identifier Descriptor内のポインタ情報を新しく作成し直したFile Entryの記録位置に更新を行う。

【0088】続いて、History Tableの再構築の要求が発生した際の処理の流れを図15に示したフローチャートに基づいて説明する。

【0089】ステップS20において、History Tableの再構築の要求が発生した場合、ステップS21において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展開さ

れる。

【0090】ステップS22において、History Table中の最後のHistory Descriptorに注目をする。ステップS23において、History Table中の全てのHistory Descriptorを見たかどうかを判定し、全てのHistory Descriptorについて処理が終っていれば処理を終了する。まだ処理が終っていない場合は、ステップS24において、注目しているHistory Descriptorのイベント種別を表すAttributeが削除かどうかを判定する。イベント種別が削除の場合は、ステップS28において注目しているHistory Descriptorを削除リストに追加する。

【0091】具体的には注目しているHistory DescriptorのFile Identifierを削除リストに登録する事を行う。ステップS29において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS23に戻り処理を繰り返す。

【0092】ステップS24においてイベント種別が削除でない場合、ステップS25において注目しているHistory Descriptorが削除リストに含まれているかを判定する。具体的には、注目しているHistory DescriptorのFile Identifierと削除リストに含まれる名前を比較する事によって行う。ステップS26において削除リストに含まれていたかを判定し、含まれていた場合は、そのままステップS29において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS23に戻り処理を繰り返す。

【0093】ステップS26において削除リストに含まれていないと判定された場合、ステップS27において抽出結果リストに追加する。具体的には注目しているHistory Descriptorをそのまま抽出結果リストにコピーする事を行う。ステップS28において、注目しているHistory Descriptorを削除リストに追加し、ステップS29において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS23に戻り処理を繰り返す。

【0094】このような処理を行う事によって、History Table中の同じファイルやディレクトリを示す重複するHistory Descriptorを削除し、更新履歴として存在した不要な情報を削除する事が可能となる。具体的には、処理が終わった段階で抽出結果リストに残っているHistory Desc

riptorが目的の情報となる。

【0095】続いて、History Tableからあるファイルやディレクトリに関する更新履歴を取得したい要求が発生した際の処理の流れを図16に示したフローチャートに基づいて説明する。

【0096】ステップS40において、History Tableからあるファイルやディレクトリに関する更新履歴を取得したい要求が発生した場合、ステップS41において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展開される。

【0097】ステップS42において、History Table中の最後のHistory Descriptorに注目をする。ステップS43において、検索対象を示す変数であるSEARCHKEYに目的のFile Identifierをセットする。ステップS44において、History Table中の全てのHistory Descriptorを調べたかどうかを判定し、全てのHistory Descriptorについて処理が終っていれば処理を終了する。まだ処理が終っていない場合は、ステップS45において、注目しているHistory DescriptorのFile Identifierが対象としているファイルあるいはディレクトリと一致するかどうかを判定する。具体的にはSEARCHKEYと注目しているHistory DescriptorのFile Identifierを比較する事によって行う。一致しない場合はステップS47において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS44に戻り処理を繰り返す。

【0098】ステップS45において、注目しているHistory DescriptorのFile Identifierが対象としているファイルあるいはディレクトリと一致する場合、ステップS46において現在注目しているHistory Descriptorは、更新履歴を取得しようとしている対象のファイルあるいはディレクトリに関する更新履歴であるものとし抽出結果としてリストアップする事になる。ステップS4

7において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS44に戻り処理を繰り返す。

【0099】ステップS47において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS44に戻り処理を繰り返す。

【0100】このような処理によって、History Tableから更新履歴を取得したいファイルあるいはディレクトリに関する全てのHistory Descriptorを簡単に抽出する事が可能となる。具体的には、処理が終わった段階の抽出結果リストに残っているHistory Descriptorが目的の情報となる。

【0101】説明してきたHistory Tableでは、アクセスしようとするファイルやディレクトリがあらかじめ分かっている場合には、History Tableに含まれる対応するHistory Descriptorを探し出すだけの処理で終わる。仮に管理情報のバックアップの対象となるファイルシステムの管理情報が全くディスクから読み出せなくなり、どのようなファイルが記録されているか、あるいはどのようなディレクトリ構造の状態であるかが不明になる事も考えられる。このような状況が発生した場合には、History Tableを再構築し残されたHistory Descriptorを参照する事によって、バックアップ情報が保持するディレクトリ階層の構造を把握する事が可能となり、バックアップ情報を介してファイルやディレクトリにアクセスしたり、管理情報の復旧をするための情報をして利用できる事になる。

【0102】ここで、UDFファイルシステムにおいて、ファイルやディレクトリを管理するFile Entryにアクセスできなかった場合に、バックアップ情報を元に管理情報を復旧する手順を図17に示すフローチャートに基づいて説明する。ステップS50において、UDFファイルシステムにおいてファイルやディレクトリを管理するFile Entryにアクセスできない事が、例えば、File Entryの読み出しが物理的に失敗したり、読み出せた場合であってもFile Entryのヘッダ情報に記録されているチェックサムやCRC情報が正しくない事によって検出された場合、ステップS51において、問題のファイルやディレクトリを示すファイル名をフルパス表現で指定してHistory Tableより対応するバックアップ情報であるHistory Descriptorを抽出する。具体的な抽出方法は既に述べた通りである。

【0103】ステップS52において、抽出したHistory Descriptorの情報からFile Entryの再生成を行うが、このFile Entr

yには、History Descriptorに含まれるAllocationDescriptorsの内容をコピーする。AllocationDescriptorsにはファイルに対応するデータの記録位置情報が格納されており、この情報が存在すればデータを読み出すための位置情報がわかるのでデータの読み出しが可能となる。再生成したFile Entryはディスク上に記録する。

【0104】ステップS53において、アクセスできなかったFile Entryの記録位置を管理しているUDFのFile Identifier DescriptorのLCBフィールドを、記録し直したFile Entryの記録位置を管理するように更新する。再生成したFile Entryを、読み出せなかったFile Entryと同じ位置に記録し直した場合は上記の更新処理を行う必要はない。

【0105】続いて、UDFファイルシステムにおいて、ディレクトリを管理するFile Entryにアクセスできなかった場合に、バックアップ情報を元に管理情報を復旧する別の手順を図18に示すフローチャートに基づいて説明する。ステップS60において、UDFファイルシステムにおいてディレクトリを管理するFile Entryにアクセスできない事が検出された場合、ステップS61において問題のディレクトリのFile Entryが記録されている位置を管理しているFile Identifier Descriptorを上位のディレクトリのディレクトリ情報から削除する。

【0106】ステップS62において、History Tableから不要なHistory Descriptorを削除するHistory Tableの再構築処理を行う。具体的な処理内容は前述した通りである。ステップS63において問題のディレクトリを示すディレクトリ名をフルパス表現で指定してHistory Tableより対応するバックアップ情報を含めそのディレクトリの下位に作成されたファイルやディレクトリに対応する全てのHistory Descriptorを抽出する。

【0107】ステップS64において、ステップS63で抽出したHistory Descriptorの情報の中から、Attribute情報を参照しディレクトリに関連するものだけ抜き出す。抜き出したディレクトリに関するHistory Descriptorが示すディレクトリを全て作成する。この処理によって、ディレクトリ構造が問題の発生する前の状態に復旧されることになる。

【0108】ステップS65において、ステップS63で抽出したHistory DescriptorのAttribute情報を参照しファイルに関連するものだけを抜き出す。抜き出したファイルに関するHist

ory DescriptorのFile Identifierを解析し、それぞれのファイルが記録されているべきディレクトリのディレクトリ情報にFile IdentifierDescriptorを追加していく。この時、対応するファイルのFile Entryの記録位置を管理するFile Identifier Descriptor内のICBには、History Descriptorで管理するLBN of FEの情報を記録する。

【0109】ファイルのFile Entryの復旧の場合と異なり、ディスク上に記録されているファイルのFile Entryには問題が全く無いので、File Identifier Descriptorからのポインタ情報を正しくセットするだけでファイルにアクセスできる事になる。このようにLBN of FEはファイルのFile Entryへのポインタ情報を繋ぎなおすために使用する。またFile Identifier DescriptorのFile Identifierには、History DescriptorのFile Identifierからパス情報を取り除いたものを記録する。この処理を全てのファイルに対応するHistory Descriptorに対して行うことによってディレクトリ構造と共にファイル構造の管理情報も復旧された事になる。なお後処理として、ディスク上にどの管理情報からも参照されない管理情報がゴミとして残る事になるので、不要な管理情報を検出して空き領域情報を更新し未使用箇所として開放しても良い。

【0110】History Descriptorで管理するバックアップ対象のファイルやディレクトリを特定するための識別情報として管理情報の記録媒体上の記録位置情報を用いる第2の実施の形態について説明を行なう。ここで、図19にPrimary History Table Descriptorの内容を示す。Primary History Table Descriptorは、History Tableを管理するための情報やバックアップ領域の情報を管理する記述子である。

【0111】Area Sizeはバックアップ領域の大きさをbyte数で表し、Uint32型として記録される。Last HD Added Timestampは、History DescriptorがHistory Tableに追加された最終日時を記録する。この情報によって、最後にHistory Descriptorを追加した日時を把握する事ができ、例えばUDFファイルシステムの管理情報との整合性を見る場合などに利用する事ができる。Last HT Updated Timestampは、History Tableを最後に再構築した日時を記録する。この情報によりHistory Tableが保持する更新履

歴がいつからのものであるかを把握する事が可能となる。

【0112】Number of History Descriptorsは、History Tableに記録されているHistory Descriptorの数を示し、Uint32型で記録される。History Table Sizeは、History Tableのサイズをbyte数で表しUint32型で記録される。History Table中の全History Descriptorの合計サイズをByte数で表す。ディスクへのHistory Descriptorの追加は論理セクタ単位で行うのが一般的なので、仮に1論理セクタのサイズが2KBとすると、このHistory Table Sizeを2KBで割り、その商がアクセスすべきバックアップ領域の最後の論理セクタから1セクタ(PHTDのサイズ)引いた箇所からみた論理セクタ数となる。

【0113】また、余りが追加を行うべき論理セクタに既にデータが記録されている量である。つまり、追加を行う場合には目的の論理セクタを一度読み出して、既に記録されている情報の前にHistory Descriptorを追加して、その論理セクタの内容をディスクに記録する事になる。

【0114】なお、図中の、RBPはRelative Byte Positionを意味し、先頭から見た対応する管理項目の開始位置を示す情報で、Lenはその管理項目の大きさをByteで表し、Field Nameは管理項目名、Contentsは、管理項目がどのような形式で記録されなければならないかということを示す。Contentsで用いられているデータ型のうち、Uint8は符号無し8bit整数、Uint16は符号無し16bit整数、Uint32は符号無し32bit整数を意味する。Stringは文字列を格納するためのデータ型、Timestampは日時情報を格納する型である。

【0115】図20にHistory Descriptorの内容を示す。History Descriptorは、論理ファイルシステムにおけるファイルやディレクトリの管理情報のバックアップ情報であり、対応するファイルやディレクトリにアクセスするための必要最低限の情報で構成されている。

【0116】File Sizeは、バックアップの対象となるファイルのファイルサイズをbyte数で示し、Uint32型で記録される。具体的には、対応するUDFファイルシステムのFile Entry内のファイルサイズの情報を示すInformation Lengthと同じ値を記録する。Modification Date and Timeは、このファイルやディレクトリが追加、変更、削除された時刻を示し、Timestamp型で記録される。バックアップ対象

がファイルの場合、対応するUDFファイルシステムのFile Entry内の変更日時を示すModification Date and Timeと同じ値を、ディレクトリの場合はFile Entry内の作成日時を示すAccess Data and Timeと同じ値を記録する。また、ファイルやディレクトリの削除を示すHistory Descriptorの場合は、実際の削除の日時を記録する。

【0117】LBN of FEは、バックアップ対象となるファイルやディレクトリのUDFファイルシステムでのFile Entryの記録位置を示すUDFパーティション内の論理ブロック番号を示し、Uint32型で記録する。Attributesは、バックアップのイベント種別およびバックアップの対象がファイルなのかディレクトリなのかをUint16型で示す。Bit 0は、バックアップ対象がファイルなのかディレクトリなのかを示し、0の場合はファイル、1の場合はディレクトリを示す。Bit 1と2は、符号無し2bitの情報として扱い、この2bitが0の場合「作成」、1の場合は「変更」、2の場合は「削除」を示す。

【0118】Length of Allocation Descriptorsは、Allocation Descriptorsフィールドの長さをbyte数で示し、Uint32型で記録する。バックアップ対象であるファイルやディレクトリのUDFファイルシステムでのFile EntryのLength of Allocation Descriptorsと同じ値を記録する。Allocation Descriptors (AD)はディスク上のデータの記録位置を管理するための管理構造であり、Uint32型のExtent Lengthと、Uint32型のExtent Positionによって構成される。ここでは、Extent LengthとExtent Positionを合わせてShort_ad型として扱うものとする。

【0119】Extent Lengthは、分断の長さをbyte数で示し、Extent Positionは分断の開始論理ブロック番号が記録される。1つのファイルであってもファイルに対応する実データはディスク上で分断されて記録する事も可能であるため、ファイルを構成する分断の数だけAllocation Descriptorsが記録されることになる。

【0120】実際には、1つのAllocation Descriptorsが8byteなので、Length of Allocation Descriptorsで示される値を8で割ることによってAllocation Descriptorsの数が求まる。このフィールドには、バックアップ対象のUDFファイルシステムでのFile Entry内のAllocat

ion Descriptorsと同じ内容を記録する。

【0121】以上のような管理情報を用いた実際の実施例を説明して行く。図21に示すようなディレクトリ階層のファイルやディレクトリがある場合、対応するUDFファイルシステムが定義するパーティション内の論理ファイルシステムの管理情報とデータ配置の様子の例を図22に示す。

【0122】パーティション内の先頭から、パーティション内の空き領域情報を管理するSpace Bitmap Descriptor (SBD)、ファイルシステムの基本管理情報であり、Rootディレクトリを管理するFile Entry (FE Root)へのポインタ情報を持つFile Set Descriptor (FSD)、File Set Descriptorが終った事を示すTerminating Descriptor (TD)が記録されている。

【0123】また、Rootディレクトリ、DATAディレクトリをそれぞれ管理するFile Entry (FE)と、各ディレクトリの下に定義されるファイルのファイル名や属性情報、そしてそのファイルあるいは下位のディレクトリを管理するFile Entry (FE)へのポインタ情報であるFile Identifier Descriptor (FID)、Rootディレクトリの下に定義されているREADME.TXTを管理するFile Entry (FE)とDATAディレクトリの下に定義される、FILE1.DATとFILE2.DATを管理するFile Entry (FE)が記録されている。

【0124】なお、History Descriptor内のLBN of FEフィールドには、このファイルやディレクトリを管理するFile Entryが記録された論理ブロック番号が記録されることになる。図の例ではRootディレクトリ、README.TXT、DATAディレクトリ、FILE1.DAT、FILE2.DATのFile Entry (FE)はそれぞれ、LBN (論理ブロック番号) 100、200、300、400、500に記録されている。

【0125】ファイルおよびディレクトリのFile Entryはそれぞれ対応するデータが記録されているディスク上の分断情報をExtentとしてAllocation Descriptorsによって管理している。例えば、図22の例においては、FILE1.DATに対応するデータはディスクでは分断して記録されており、Extent 4および5によって構成されている。またFILE2.DATに対応するディスク上のデータは連続的に記録されており、Extent 6によって構成されている。

【0126】このような状況のディレクトリ階層に対応するHistory Tableの例を図23に示す。

この図では、一番下にPrimary History Table Descriptorが配置されており、History Tableの最後尾は図の一番上に相当する。この例では、Rootディレクトリ

(¥)、README.TXT、¥DATAディレクトリ、FILE1.DAT、FILE2.DATの順番で作成された様子を示している。一番下の枠を除いて1つの枠が1つのHistory Descriptorに対応し、全てのHistory DescriptorをまとめてHistory Tableを構成する。

【0127】図24に図23に示す状態から、DATAディレクトリの下にFILE3.DATというファイルが作成(追加)された場合の様子を示す。FILE3.DATがUDFファイルシステムで作成されると、それに対応するHistory DescriptorがHistory Tableの最後尾(ディスク上では最前部)に追加される。

【0128】この際、History DescriptorのLBN of FEにはFILE3.DATを管理するFile Entryの記録されたディスク上での位置情報が記録され、Timestampにはファイルの作成された日時、Attributeにはファイルおよび作成を示す0000h、File SizeにはFILE3.DATのファイルサイズを、Length of Allocation Descriptorsには、FILE3.DATに対応するデータのディスク上での分断数を8倍した値を、そしてAllocation Descriptors(AD)にはそれぞれの分断に関する位置情報を記録する事になる。ここでは、FILE3.DATはExtent7のみによって構成されている(連続して記録されている)ことになる。

【0129】図25に、図24で示された状態から、さらにDATAディレクトリの下にFILE2.DATの内容が変更された場合の様子を示す。UDFファイルシステムにおいてFILE2.DATのディスク上の位置が変更になったり、内容が変更になった場合、History Tableの最後尾(ディスク上では最前部)に、対応するHistory Descriptorを追加する。この際、Attributeにはファイルおよび変更を示す0002hを記録する。また、FILE2.DATのディスク上の位置が変更となり、Allocation Descriptors(AD)が管理する分断がExtent8に変更になる。

【0130】このFILE2.DATの変更に関するHistory Descriptorが追加になった時点で、History Table内に既に存在している、FILE2.DATを作成した際のHistory Descriptorが古い情報となり、更新履歴情報となる。

【0131】図26に、図25で示された状態から、さらにDATAディレクトリの下にFILE1.DATを削除した場合の様子を示す。UDFファイルシステムにおいてFILE1.DATが削除された場合、History Tableの最後尾(ディスク上では最前部)に対応するHistory Descriptorを追加する。この際、Attributeにはファイルおよび削除を示す0004hを記録し、Allocation Descriptors(AD)には、削除する直前のデータの位置情報を記録する。削除されるデータであるため直前の記録位置を記録しないようにしても良い。このFILE1.DATの削除に関するHistory Descriptorが追加になった時点で、History Table内に既に存在しているFILE1.DATを作成した際のHistory Descriptorが古い情報となり、更新履歴情報となる。

【0132】なお、図24乃至26においてイベント種別が作成、変更、削除について説明したが、ファイルシステム上では例えば、ファイルやディレクトリのコピーや移動あるいは名称変更なども起きる。これらのイベントは基本的に前述の作成、変更、削除を組み合わせることによって更新情報を表現する事が可能である。例えば、あるファイルの移動であれば、移動元のファイルに対応する削除を表すHistory Descriptorを作成し、移動先のファイルに対応する作成を表すHistory Descriptorを作成する。また、同様に名称変更でも、名称を変更する前のファイルに対応する削除を表すHistory Descriptorを作成し、名称変更後のファイルに対応する作成を表すHistory Descriptorを追加すれば良いことになる。

【0133】ここで、History Tableを再構築する場合の様子を図27に示す。History Tableには、UDFファイルシステムにおいてファイルに関する作成、変更、削除、またディレクトリに関する作成、削除と言ったイベントが発生する度にHistory Descriptorが単純に追加になっていく。History Descriptorを記録する領域は決まっているので、状況に応じてHistory Tableから不要な情報を削除しHistory Tableを再構築する事も必要となる。

【0134】この場合は、History Tableの最後部(ディスク上では最前部)のHistory Descriptorから順番に見て行き、同一のファイルやディレクトリに関する情報つまり更新履歴に相当するHistory DescriptorをHistory Tableから削除する。また、Attributeが削除のHistory Descriptorに関しても同様に削除する。

【0135】図の例では、左側のHistory Ta

bleから更新履歴に相当する「FILE2.DAT作成」と「FILE1.DAT作成」と、「FILE1.DAT削除」のHistory Descriptorを削除し、右のHistory Tableのように再構築する。この際、残ったHistory DescriptorのAttributeは全て作成に変更する事になる。

【0136】ここで、処理の詳細をフローチャートで説明する。図28にUDFファイルシステムにおいてファイルやディレクトリの作成、変更、削除と言ったイベントが発生した場合の処理の流れを示す。

【0137】ステップS70において、ファイルやディレクトリの作成、変更、削除イベントが発生すると、ステップS71においてPrimary History Table Descriptorを読み出し、History Tableの最後部（ディスク上では最前部）を把握する。

【0138】具体的には、History Table中のHistory TableSizeからHistory Tableの最後尾の位置を把握する事が可能となる。ステップS72において、ファイルやディレクトリに対応するHistory DescriptorをHistory Tableの最後に追加する。この際History Descriptorには、図21に示すように、ファイルやディレクトリを管理するFile Entryの記録位置をLBN of FEに、発生したイベントの種類およびファイルかディレクトリなのかをAttribute、イベントが発生した日時、イベントが発生した対象のUDFファイルシステムにおけるファイルの管理するディスク上の分断を管理する形で、Allocation Descriptorsが記録される。ステップS73において、Primary History Table Descriptor内の最後尾（ディスク上では最前部）にHistory Descriptorを追加した時刻を示すLast HD Added Timestampと、History Descriptor数、History Table Sizeを更新して処理を終了する。UDFにおけるファイルに関するイベントが発生した際の処理は以上のように、単純なHistory Descriptorの追加処理となる。

【0139】ファイルの場合とは異なり、バックアップ対象がディレクトリの場合は、ディレクトリの存在のみをHistory Descriptorで管理することにし、File SizeやAllocation Descriptorsを記録しないようにしても良い。

【0140】このフローチャートでは、UDFのファイルシステムにおいてファイルやディレクトリの作成、変更、削除と言ったイベントが発生する度に対応するH

story DescriptorをHistory Tableに追加する説明になっているが、例えば電源投入時やディスクを装着した時にPrimary History Table Descriptorを読み出し、電源を切断する時やディスクを排出する時などにその期間中に発生した全てのイベントに対応するHistory DescriptorをまとめてHistory Tableに追加しPrimary History Table Descriptorを更新し記録しても良い。つまりHistory Descriptorをメモリ上に保持しておき、あるタイミングで一度にディスク上のHistory Tableを更新しても構わないものとする。

【0141】このように、ファイルやディレクトリに関するイベントが発生する度にHistory Descriptorを追記する訳である。UDFにおいてファイルやディレクトリにアクセスする場合、まずFile Identifier Descriptorにアクセスしディレクトリ内に記録されているファイルやディレクトリ名を把握する。アクセスしたいファイルやディレクトリに対応するFile Identifier Descriptorの情報より、対応するFile Entryの記録位置を把握し、File Entryにアクセスする。

【0142】File Entryにはファイルやディレクトリの実体の記録位置が管理されているので、この情報を元に実体にアクセスする訳である。このときファイルやディレクトリにアクセスしようとしたが、それらを管理するFile Entryにアクセスできない非常時に、バックアップ情報であるHistory Tableにアクセスする手段に関して図29のフローチャートを用いて説明する。

【0143】ステップS80において、バックアップ情報であるHistory Tableへのアクセス要求が発生すると、ステップS81において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置（ディスク上では最前部）を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展開される。ステップS82において、History Table中の最後（時間的には最新）のHistory Desc

riptorに注目をする。

【0144】ステップS83において、今アクセスしたファイルあるいはディレクトリのFile Entryの記録されている位置情報(LBN)と注目しているHistory DescriptorのLBN of FEが一致するかどうかを判定する。一致しない場合はステップS84において全てのHistory Descriptorをサーチしたかどうかを判定する。全てのHistory Descriptorをサーチした場合は、探そうとしているバックアップ情報が見つからないという事になり、ステップS87においてエラー処理を行ない処理を終了する。ステップS84においてまだ全てのHistory Descriptorをサーチし終わっていない場合は、ステップS85において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS83に戻り処理を繰り返す。

【0145】ステップS83においてLBN of FEが一致すると判定された場合は、ステップS86において注目しているHistory Descriptorのイベント種別を示すAttributeが削除であるかどうかを判定する。イベント種別が削除の場合、探そうとしているファイルあるいはディレクトリのバックアップ情報が削除状態を示すものであるためステップS87においてエラー処理して処理を終了する。ただし、ここで、ファイルやディレクトリの削除した時点での情報が取得したいのであれば、エラー処理を行わず抽出したHistory Descriptorを利用しても構わない。ステップS86においてイベント種別が削除以外であれば、注目しているHistory Descriptorが探し出すべきバックアップ情報であり、サーチ処理を終了する。探し出したHistory Descriptor中のAllocation Descriptorsの情報をを用いてファイルやディレクトリのデータにアクセスが可能となるわけである。

【0146】このように、ある特定のファイルやディレクトリのUDFファイルシステム管理情報にアクセスできない場合に、History Tableから対応する最新のHistory Descriptorを抽出し、その情報から対応するデータの記録位置を把握しデータにアクセスする事を可能とした。また、この情報を利用して主の管理情報であるUDFの管理情報を再構築する事も可能である。

【0147】例えば、あるファイルを管理するUDFのFile Entryが読み出せなくなった場合は、対応するバックアップ情報に相当するHistory Descriptorの情報を元にFile EntryをUDFのパーティション内に記録し直す。

【0148】File Entryの情報としては、作

成時刻はFile Entryを作成し直した時刻、属性情報は標準的な情報にセットされ、問題の発生する前の状態とは必ずしも同じでは無いが、データにアクセスするための必要最低限の情報であるデータの記録位置情報に関しては完全に復活できるわけである。復旧したUDFのFile Entryの記録位置が問題のあった元のFile Entryの記録位置と異なる場合、File Entryの記録位置を管理しているUDF管理情報のFile Identifier Descriptor内のポインタ情報を新しく作成し直したFile Entryの記録位置に更新を行う。

【0149】続いて、History Tableの再構築の要求が発生した際の処理の流れを図30に示したフローチャートに基づいて説明する。

【0150】ステップS90において、History Tableの再構築の要求が発生した場合、ステップS91において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展開される。

【0151】ステップS92において、History Table中の最後のHistory Descriptorに注目をする。ステップS93において、History Table中の全てのHistory Descriptorを見たかどうかを判定し、全てのHistory Descriptorについて処理が終っていれば処理を終了する。まだ処理が終わっていない場合は、ステップS94において、注目しているHistory Descriptorのイベント種別を表すAttributeが削除かどうかを判定する。イベント種別が削除の場合は、ステップS98において注目しているHistory Descriptorを削除リストに追加する。具体的には注目しているHistory DescriptorのLBN of FEを削除リストに登録する事を行う。ステップS99において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS93に戻り処理を繰り返す。

【0152】ステップS94においてイベント種別が削

除でない場合、ステップS95において注目しているHistory Descriptorが削除リストに含まれているかを判定する。具体的には、注目しているHistory DescriptorのLBN of FEと削除リストに含まれるLBN of FEを比較する事によって行う。ステップS96において削除リストに含まれていたかを判定し、含まれていた場合は、そのままステップS99において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS93に戻り処理を繰り返す。

【0153】ステップS96において削除リストに含まれていないと判定された場合、ステップS97において抽出結果リストに追加する。具体的には注目しているHistory Descriptorをそのまま抽出結果リストにコピーする事を行う。ステップS98において、注目しているHistory Descriptorを削除リストに追加し、ステップS99において注目しているHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS93に戻り処理を繰り返す。

【0154】このような処理を行う事によって、History Table中の同じファイルやディレクトリを示す重複するHistory Descriptorを削除し、更新履歴として存在した不要な情報を削除する事が可能となる。具体的には、処理が終わった段階で抽出結果リストに残っているHistory Descriptorが目的の情報となる。

【0155】続いて、History Tableからあるファイルやディレクトリに関する更新履歴を取得したい要求が発生した際の処理の流れを図31に示したフローチャートに基づいて説明する。

【0156】ステップS100において、History Tableからあるファイルやディレクトリに関する更新履歴を取得したい要求が発生した場合、ステップS101において、Primary History Table Descriptorを読み出し、History Tableの大きさと、History Table中のHistory Descriptorの数を把握してHistory Tableをディスクから読み出す。

【0157】具体的には、History Table中のNumber of History DescriptorsによってHistory Table中のHistory Descriptorの数を、History Table SizeからHistory Tableの最後尾の位置を把握する事が可能となり、読み出されたHistory Tableは制御部のメモリ上でHistory Descriptor毎に展

開される。

【0158】ステップS102において、History Table中の最後のHistory Descriptorに注目をする。ステップS103において、検索対象を示す変数であるSEARCHKEYに目的のLBN of FEをセットする。LBN of FEは、検出対象のファイルやディレクトリを管理するFile Identifier Descriptorによって把握する事が可能である。ステップS104において、History Table中の全てのHistory Descriptorを調べたかどうかを判定し、全てのHistory Descriptorについて処理が終わっていれば処理を終了する。

【0159】まだ処理が終わっていない場合は、ステップS105において、注目しているHistory DescriptorのLBN of FEが対象としているファイルあるいはディレクトリと一致するかどうかを判定する。具体的にはSEARCHKEYと注目しているHistory DescriptorのLBN of FEを比較する事によって行う。一致しない場合はステップS107において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS104に戻り処理を繰り返す。

【0160】ステップS105において、注目しているHistory DescriptorのLBN of FEが対象としているファイルあるいはディレクトリと一致する場合、ステップS106において現在注目しているHistory Descriptorは、更新履歴を取得しようとしている対象のファイルあるいはディレクトリに関する更新履歴であるものとし抽出結果としてリストアップする事になる。ステップS107において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS104に戻り処理を繰り返す。

【0161】ステップS107において注目するHistory DescriptorをHistory Table中の1つ前のHistory Descriptorに変更し、ステップS104に戻り処理を繰り返す。

【0162】このような処理によって、History Tableから更新履歴を取得したいファイルあるいはディレクトリに関する全てのHistory Descriptorを簡単に抽出する事が可能となる。具体的には、処理が終わった段階の抽出結果リストに残っているHistory Descriptorが目的の情報となる。

【0163】説明してきたHistory Tableでは、アクセスしようとするファイルやディレクトリが

あらかじめ分かっている場合には、History Tableに含まれる対応するHistory Descriptorを探し出すだけの処理で終わる。仮に管理情報のバックアップの対象となるファイルシステムの管理情報が全くディスクから読み出せなくなり、どのようなファイルが記録されているか、あるいはどのようなディレクトリ構造の状態であるかが不明になる事も考えられる。このような状況が発生した場合には、History Tableを再構築し残されたHistory Descriptorを参照する事によって、バックアップ情報を介してファイルの実体にアクセスしたり、管理情報の復旧をするための情報をして利用できる事になる。

【0164】ここで、UDFファイルシステムにおいて、ファイルやディレクトリを管理するFile Entryにアクセスできなかった場合に、バックアップ情報を元に管理情報を復旧する手順を図32に示すフローチャートに基づいて説明する。

【0165】ステップS110において、UDFファイルシステムにおいてファイルやディレクトリを管理するFile Entryにアクセスできない事が、例えば、File Entryの読み出しが物理的に失敗したり、読み出した場合であってもFile Entryのヘッダ情報に記録されているチェックサムやCRC情報が正しくない事によって検出された場合、ステップS111において、問題のファイルやディレクトリのFile Entryの記録位置を示すLBN of FEで指定してHistory Tableより対応するバックアップ情報であるHistory Descriptorを抽出する。具体的な抽出方法は既に述べた通りである。ステップS112において、抽出したHistory Descriptorの情報からFile Entryの再生成を行うが、このFile Entryには、History Descriptorに含まれるAllocation Descriptorsの内容をコピーする。

【0166】Allocation Descriptorsにはファイルに対応するデータの記録位置情報が格納されており、この情報が存在すればデータを読み出すための位置情報がわかるのでデータの読み出しが可能となる。再生成したFile Entryはディスク上に記録する。ステップS113において、アクセスできなかったFile Entryの記録位置を管理しているUDFのFile Identifier DescriptorのICBフィールドを、記録し直したFile Entryの記録位置を管理するように更新する。再生成したFile Entryを、読み出せなかったFile Entryと同じ位置に記録し直した場合は上記の更新処理を行う必要はない。

【0167】第2の実施形態では、History D

escriptorで管理するバックアップ対象のファイルやディレクトリを特定するための識別情報として管理情報の記録媒体上の記録位置情報を用い、ファイル名やディレクトリ名は管理していない。よって、バックアップ情報単独でファイルの実体にアクセスする事は可能であるが、そのファイル名やディレクトリ階層を再構築する事はできない。つまり、通常はUDFにおけるファイルやディレクトリの名前と対応するFile Entryの記録位置を管理するFile Identifier Descriptorの情報にアクセスできる事が前提となる。よってFile Identifier Descriptorにアクセスできなくなると、バックアップ情報だけではファイル名やディレクトリ階層の復旧ができなくなってしまう。

【0168】この問題を解決するために、図33に示すようにディレクトリを管理するFile Entryとファイルやディレクトリの名前と対応するFile Entryの記録位置等を管理するFile Identifier Descriptorをディレクトリ構造用管理情報領域に記録する。この領域に記録されている管理情報にアクセスする事によってディスクに記録されているファイルやディレクトリ構造を容易に把握する事が可能となる。ファイルの実体を管理するFile Entryをこの領域に記録しないため、大量のファイルを作成しても領域として必要な大きさを抑える事が可能である。これは、File Identifier Descriptorのデータ量が少ないためである。

【0169】例えば1つのファイルを管理するFile Entryはディスク上の1論理ブロック(2KB)を占有してしまうが、File Identifier Descriptorは12文字のファイル名の場合52byteしか占有しない。よって同じ2KBで管理できるFile Identifier Descriptorの数は約39ファイル分となる。

【0170】例えば、図33に示すように上記管理情報用の領域をそのまま後続する領域全体をコピーしその領域を1つのファイルとして管理する事が考えられる。このディレクトリ構造管理情報領域のバックアップファイルに特定の名前をつける事によって、ディレクトリ構造を管理するUDFの管理情報のバックアップにアクセスする事が可能となる。別の手段としてこの管理情報用のバックアップをファイルとして管理するのではなく、UDFのパーティションの外の記録位置が固定された専用領域に記録しても良い。いずれにしても、File Identifier Descriptorが2重化されているため、万が一File Identifier Descriptorにアクセスできないような問題が発生した場合には、このディレクトリ構造の管理情報のバックアップにアクセスする事によって問題を解決する事が可能である。

【0171】本発明第1および第2の実施の形態では、History Tableをディスク上に確保された専用のバックアップ領域に記録したが、この情報を専用領域に記録せずにバックアップの対象となるファイルシステム内で普通のファイルとして記録する事も考えられる。このような構成を取る事によって、バックアップ用の専用領域を用意する必要が無く、ファイルとして記録されるのでバックアップ情報であるHistory Tableをディスク上の任意の箇所に記録する事が可能となる。

【0172】また、History Tableの内容を不揮発性の半導体メモリに格納する事も考えられる。このように、データ自体が記録されている記録媒体と異なる媒体にHistory Tableを記録する場合は、History Tableと対応するファイルシステムが記録されているメディアを識別するための情報と共に記録する事によって、ディスクメディアとHistory Tableの対応を取る事が可能となる。

【0173】

【発明の効果】本発明によれば、論理ファイルシステムの管理情報を多重する機能が無いようなファイルシステムであっても、ファイルに関する作成、変更、削除、ディレクトリに関する作成、削除と言ったイベントが発生する度に、対応するファイルやディレクトリにアクセスするための識別情報と、ファイルやディレクトリの管理情報を復旧するための必要最低限の情報とで構成されるHistory DescriptorをHistory Tableに単純な処理である追加処理を行う。また、History DescriptorをHistory Tableを記録する領域の後方から前方方向へ順番に記録することによって、History Table内のHistory Descriptorへのアクセスを容易に実現させる事になる。

【0174】この事によって、論理ファイルシステムの管理情報が読み出せなくなった場合にこのバックアップ情報であるHistory Tableにアクセスする事によって対応するファイルやディレクトリにアクセスする事と管理情報の復旧が可能となる。

【0175】また、バックアップ情報であるHistory Tableは、ファイルやディレクトリに関する作成、変更、削除と言ったイベントが発生する度に追加されるものなので、ファイルやディレクトリの更新履歴情報としても用いる事が可能となる。

【図面の簡単な説明】

【図1】本発明のディスク管理方法の実施形態におけるブロック図を示す説明図である。

【図2】本発明のディスク管理方法の実施形態においてUDFのパーティションとバックアップ領域の関係を示す説明図である。

【図3】本発明のディスク管理方法の実施形態において

バックアップ領域の内容を示す説明図である。

【図4】本発明のディスク管理方法の第1実施形態におけるPrimary History Table Descriptorを示す説明図である。

【図5】本発明のディスク管理方法の第1実施形態におけるHistory Descriptorを示す説明図である。

【図6】本発明のディスク管理方法の第1実施形態におけるディレクトリ階層の例を示す説明図である。

10 【図7】本発明のディスク管理方法の第1実施形態における図6に対応するUDF管理情報とデータのディスク上での配置の例を示す説明図である。

【図8】本発明のディスク管理方法の第1実施形態における図7に対応するHistory Tableの一例を示す説明図である。

【図9】本発明のディスク管理方法の第1実施形態においてファイルが追加になった場合のHistory Tableの更新の様子を示す説明図である。

20 【図10】本発明のディスク管理方法の第1実施形態においてファイルが変更になった場合のHistory Tableの更新の様子を示す説明図である。

【図11】本発明のディスク管理方法の第1実施形態においてファイルが削除になった場合のHistory Tableの更新の様子を示す説明図である。

【図12】本発明のディスク管理方法の第1実施形態においてHistory Tableを再構築する様子を示す説明図である。

30 【図13】本発明のディスク管理方法の第1実施形態におけるファイルやディレクトリの作成、変更、削除と言ったイベントが発生した際の処理の手順を示すフローチャートである。

【図14】本発明のディスク管理方法の第1実施形態におけるバックアップ情報であるHistory Tableにアクセスする際の処理の手順を示すフローチャートである。

【図15】本発明のディスク管理方法の第1実施形態におけるバックアップ情報であるHistory Tableを再構築際の処理の手順を示すフローチャートである。

40 【図16】本発明のディスク管理方法の第1実施形態におけるバックアップ情報であるHistory Tableから特定のファイルあるいはディレクトリに関する更新履歴を把握する処理の手順を示すフローチャートである。

【図17】本発明のディスク管理方法の第1実施形態におけるバックアップ情報であるHistory Tableを用いてUDFファイルシステムにおけるファイルを管理するFile Entryを復旧する手順を示すフローチャートである。

50 【図18】本発明のディスク管理方法の第1実施形態に

におけるバックアップ情報であるHistory Tableを用いてUDFファイルシステムにおけるディレクトリを管理するFile Entryを復旧する手順を示すフローチャートである。

【図19】本発明のディスク管理方法の第2実施形態におけるPrimary History Table Descriptorを示す説明図である。

【図20】本発明のディスク管理方法の第2実施形態におけるHistory Descriptorを示す説明図である。

【図21】本発明のディスク管理方法の第2実施形態におけるディレクトリ階層の例を示す説明図である。

【図22】本発明のディスク管理方法の第2実施形態における図6に対応するUDF管理情報とデータのディスク上での配置の例を示す説明図である。

【図23】本発明のディスク管理方法の第2実施形態における図7に対応するHistory Tableの一例を示す説明図である。

【図24】本発明のディスク管理方法の第2実施形態においてファイルが追加になった場合のHistory Tableの更新の様子を示す説明図である。

【図25】本発明のディスク管理方法の第2実施形態においてファイルが変更になった場合のHistory Tableの更新の様子を示す説明図である。

【図26】本発明のディスク管理方法の第2実施形態においてファイルが削除になった場合のHistory Tableの更新の様子を示す説明図である。

【図27】本発明のディスク管理方法の第2実施形態においてHistory Tableを再構築する様子を示す説明図である。

【図28】本発明のディスク管理方法の第2実施形態におけるファイルやディレクトリの作成、変更、削除と言ったイベントが発生した際の処理の手順を示すフローチャートである。

*

*【図29】本発明のディスク管理方法の第2実施形態におけるバックアップ情報であるHistory Tableにアクセスする際の処理の手順を示すフローチャートである。

【図30】本発明のディスク管理方法の第2実施形態におけるバックアップ情報であるHistory Tableを再構築際の処理の手順を示すフローチャートである。

10 【図31】本発明のディスク管理方法の第2実施形態におけるバックアップ情報であるHistory Tableから特定のファイルあるいはディレクトリに関する更新履歴を把握する処理の手順を示すフローチャートである。

【図32】本発明のディスク管理方法の第2実施形態におけるバックアップ情報であるHistory Tableを用いてUDFファイルシステムにおけるファイルやディレクトリを管理するFile Entryを復旧する手順を示すフローチャートである。

20 【図33】本発明のディスク管理方法の第2実施形態におけるディレクトリ構造を管理するUDF管理情報の2重化の様子の説明図である。

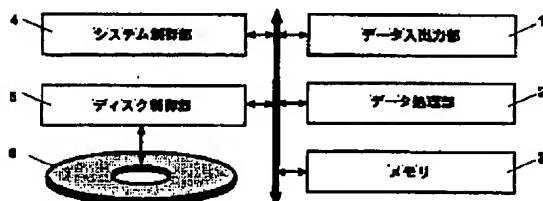
【図34】従来技術における論理ファイルシステムの管理情報とデータの記録される領域の様子の説明図である。

【図35】従来技術における論理ファイルシステムの管理情報とデータの記録される領域が別れているファイルシステムの様子の説明図である。

【符号の説明】

- 1 データ入出力部
2 データ処理部
3 メモリ
4 システム制御部
5 ディスク制御部
6 ディスク

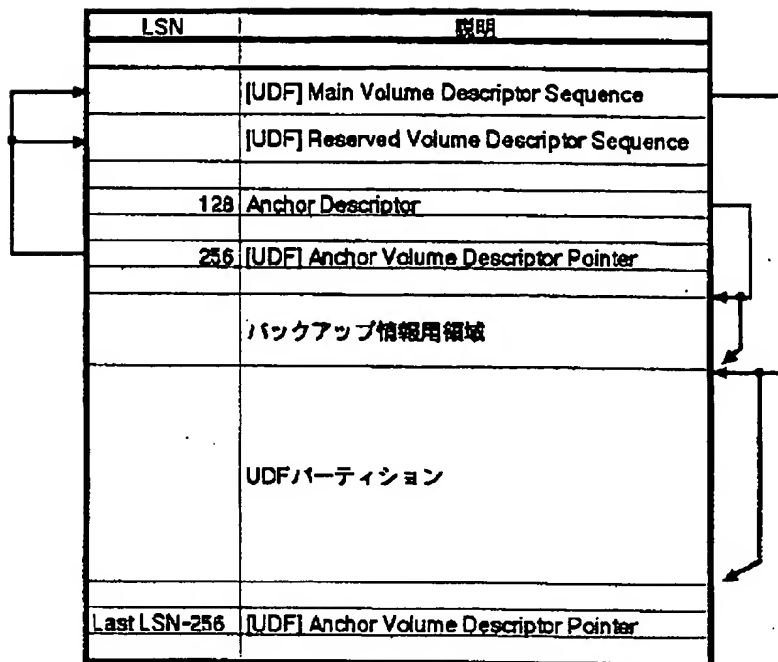
【図1】



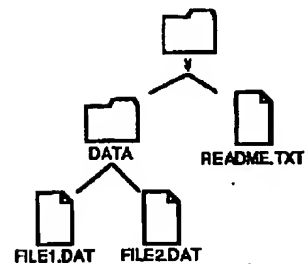
【図4】

Primary History Table Descriptor			
BBP	Len	Field Name	Contents
0	4	Area Size	UInt32
4	12	Last HD Added Timestamp	Timestamp
16	12	Last HT Updated Timestamp	Timestamp
28	4	Number of History Descriptors	UInt32
32	4	History Table Size	UInt32
36	2012	Reserved	#00 bytes

【図2】



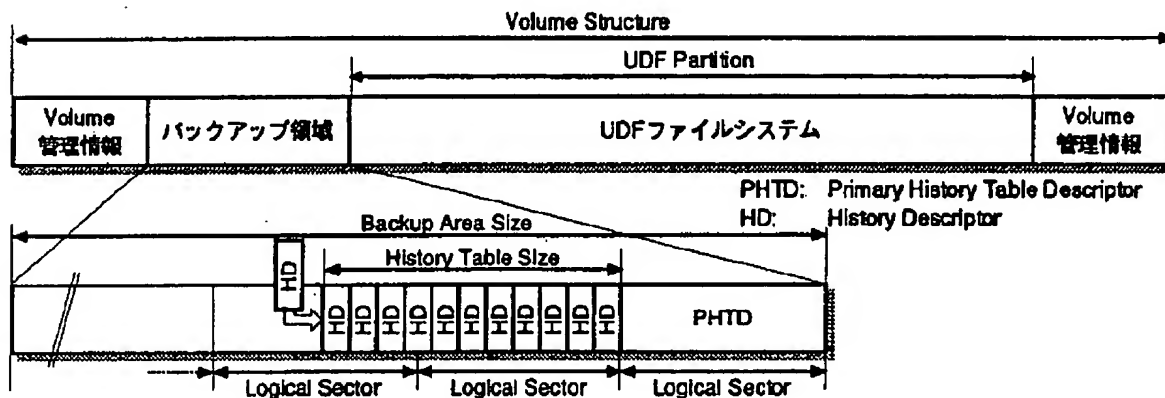
【図6】



【圖 19】

Primary History Table Descriptor				
Offset	Len	Field Name	Contents	
0	4	Area Size	UInt32	
4	12	Last HD Added Timestamp	Timestamp	
16	12	Last HT Updated Timestamp	Timestamp	
28	4	Number of History Descriptors	UInt32	
32	4	History Table Size	UInt32	
36	2012	Reserved	#00 bytes	

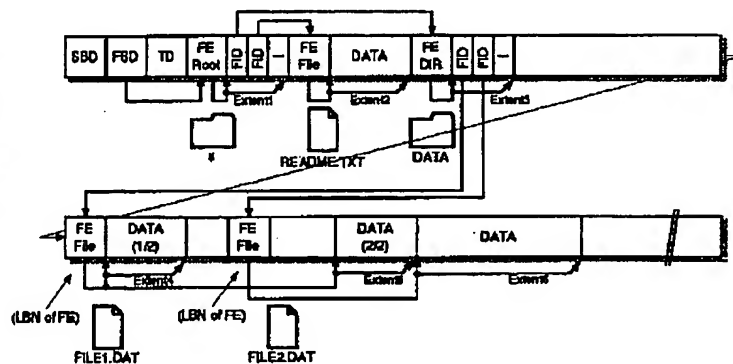
【图 3】



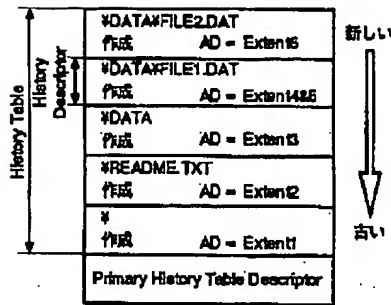
【圖5】

History Descriptor			
RBPF	Len	Field Name	Contents
0	4	File Size	UInt32
4	12	Modification Date and Time	Timestamp
16	4	LBN of FE	UInt32
20	2	Attributes	UInt6
22	2	Length of File Identifier	UInt6 and LF
24	4	Length of Allocation Descriptors	UInt32-LAD
28	LF	File Identifier	String
28+LF	LP	Padding	#00 bytes
28+LF+LAD	LAD	Allocation Descriptors	Short ad

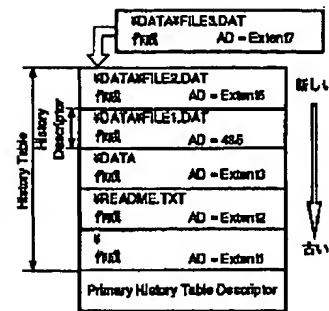
【图7】



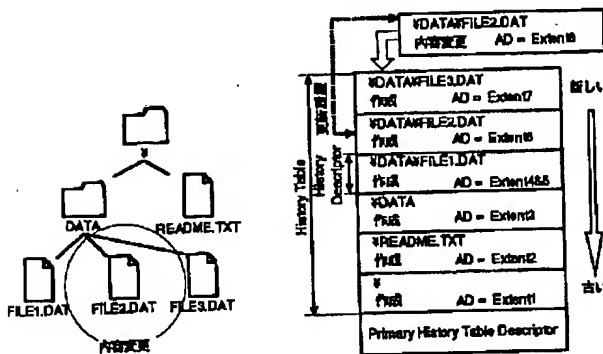
【図8】



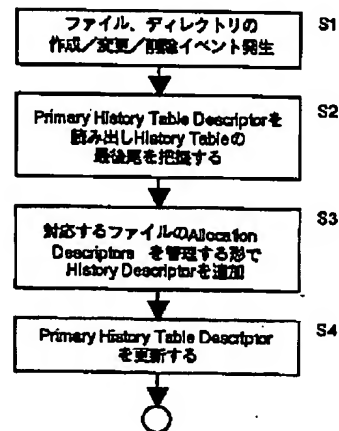
【図9】



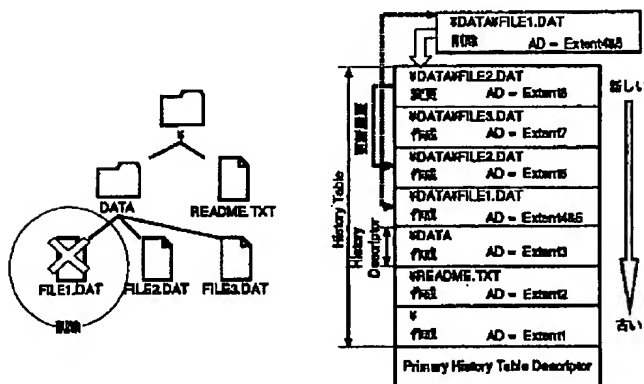
【図10】



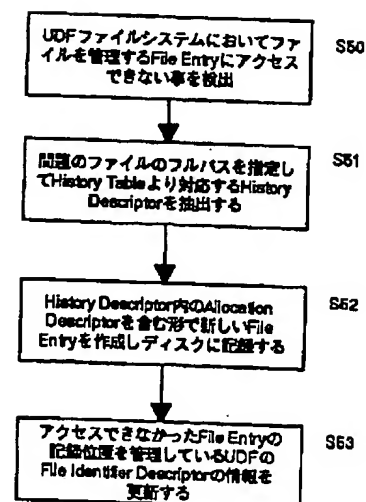
【図13】



【図11】



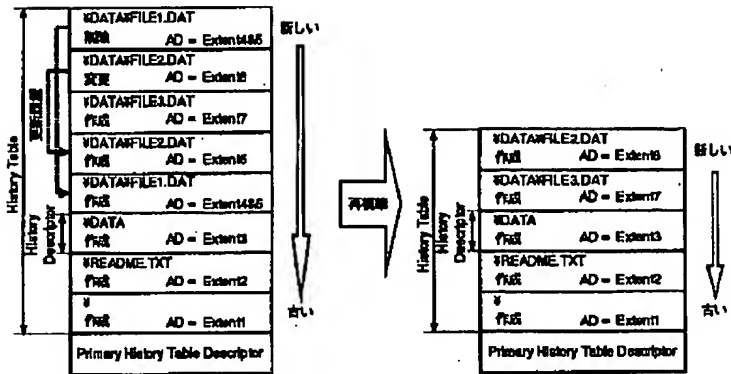
【図17】



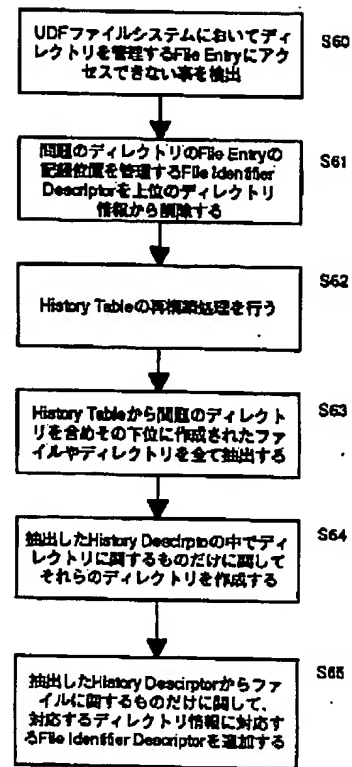
【図20】

History Descriptor			
RBP	Len	Field Name	Contents
0	4	File Size	UInt32
4	12	Modification Date and Time	Timestamp
16	4	LBN of FE	UInt32
20	4	Attributes	UInt32
24	4	Length of Allocation Descriptor	UInt32
28	L AD	Allocation Descriptors	Short ad

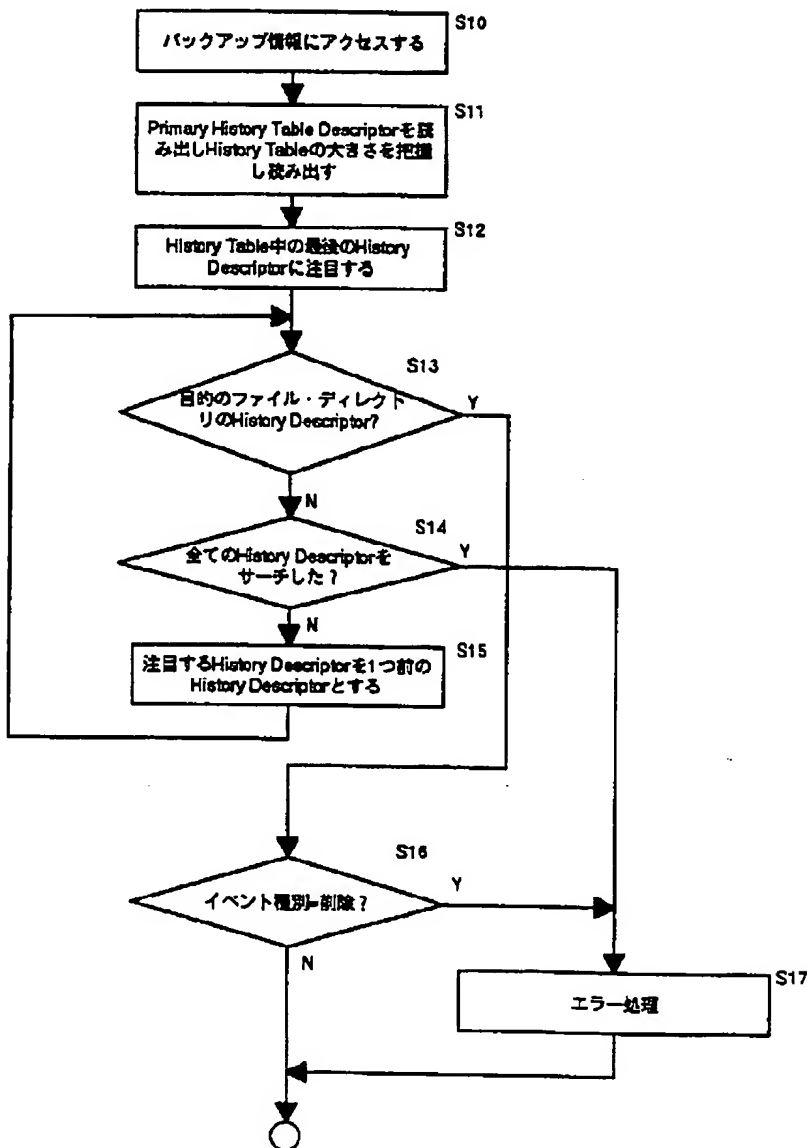
【図12】



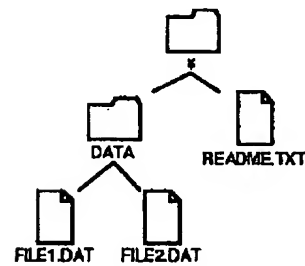
【図18】



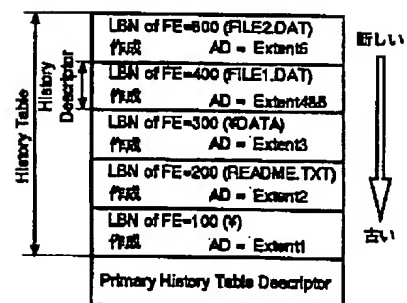
【図14】



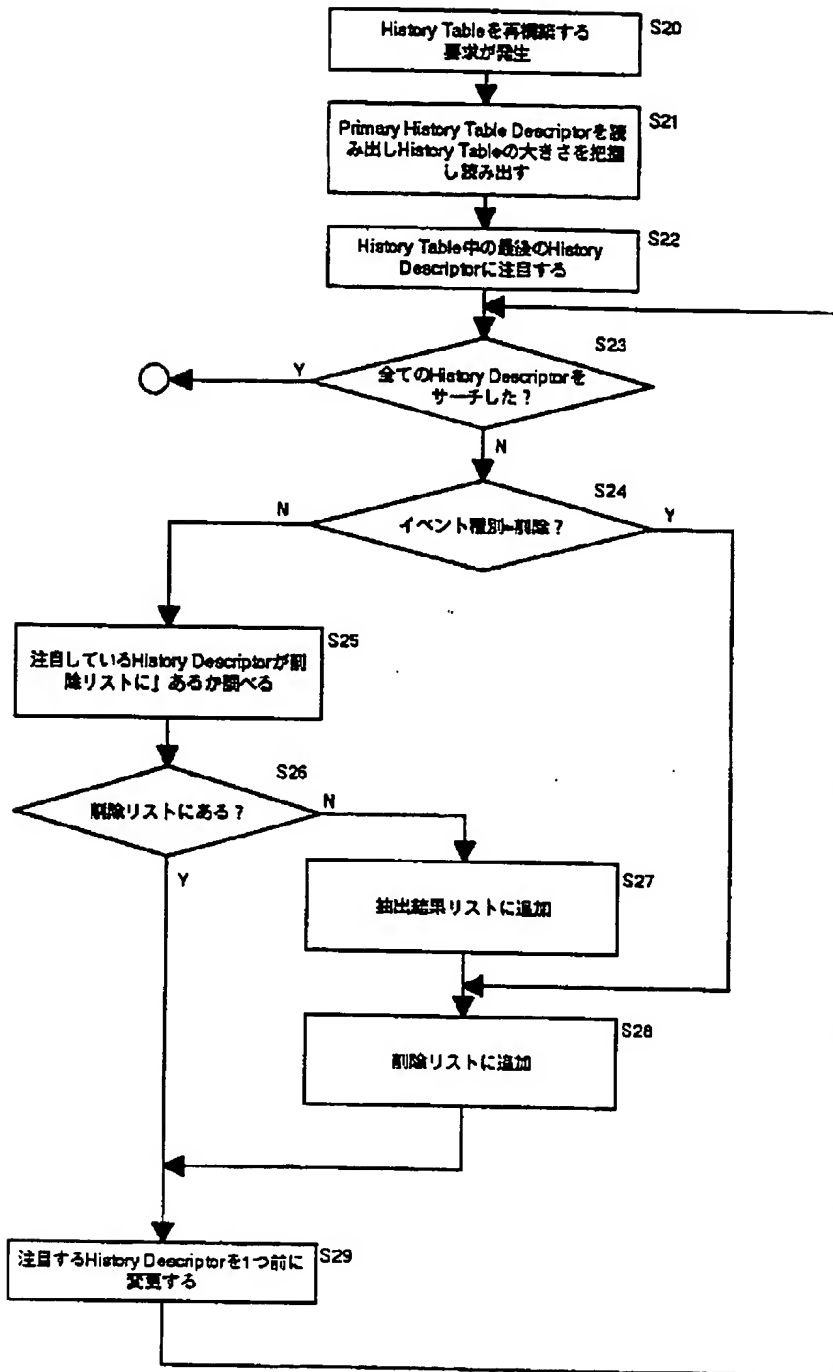
【図21】



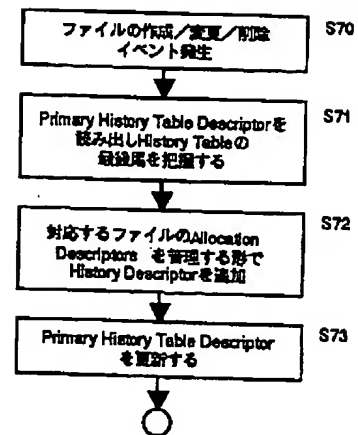
【図23】



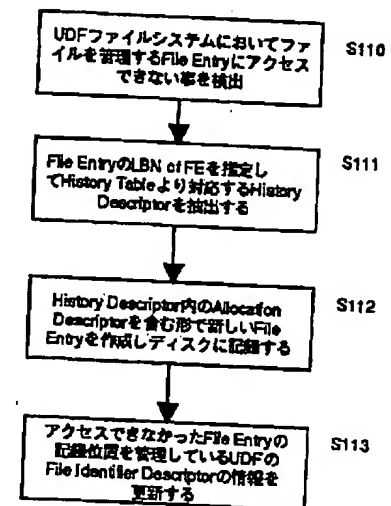
【図15】



【図28】



【図32】

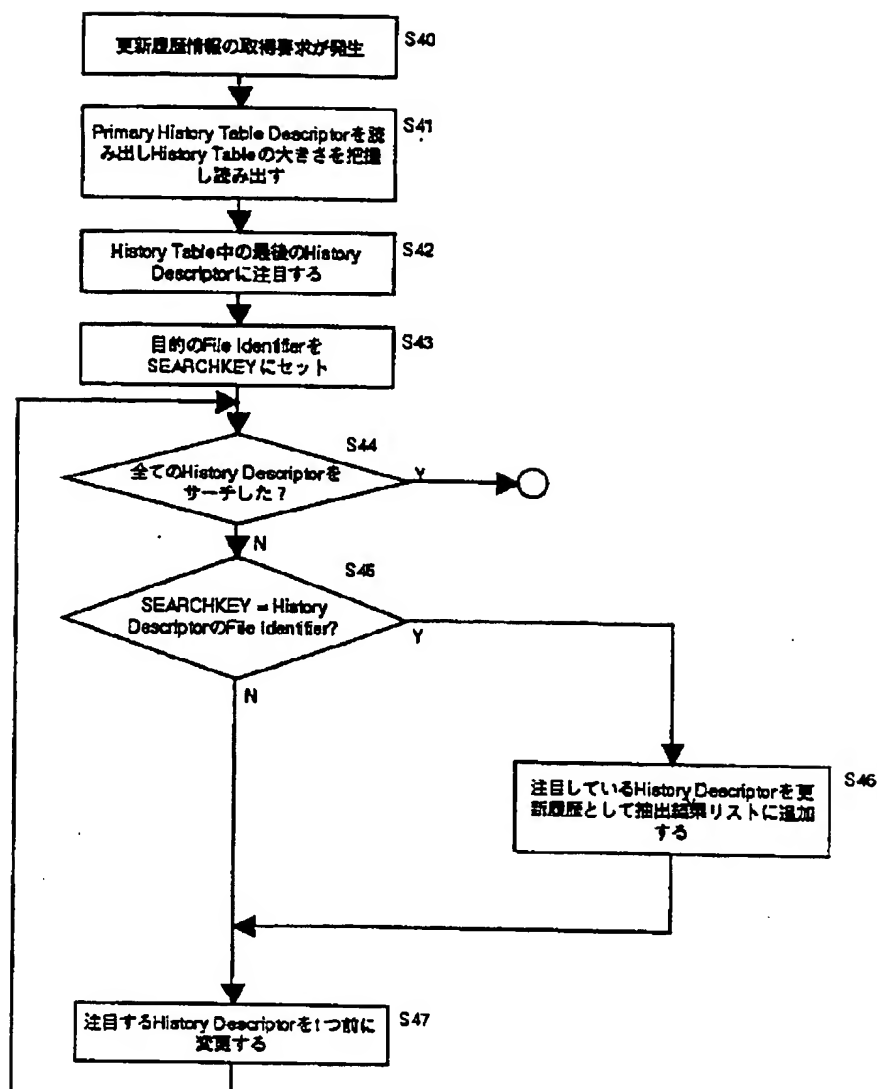


【図34】

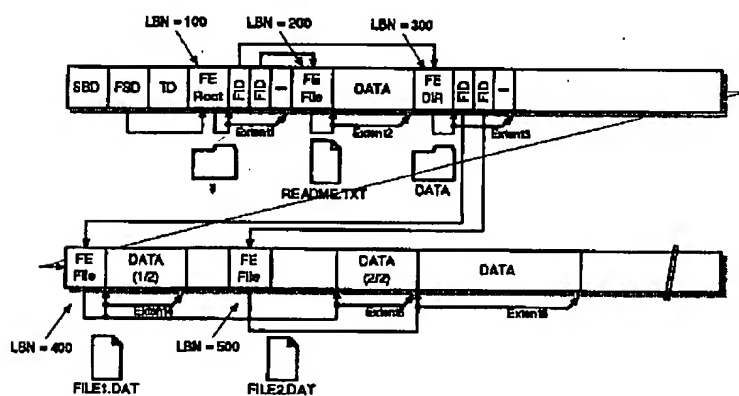
【図35】



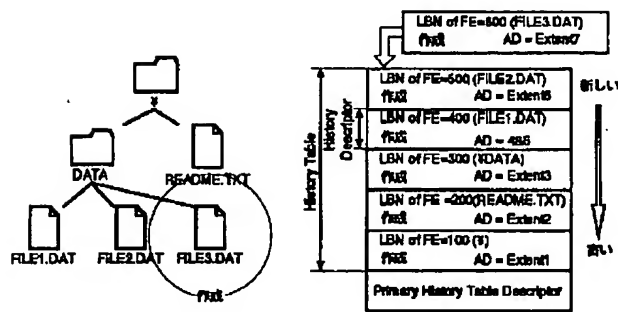
【図16】



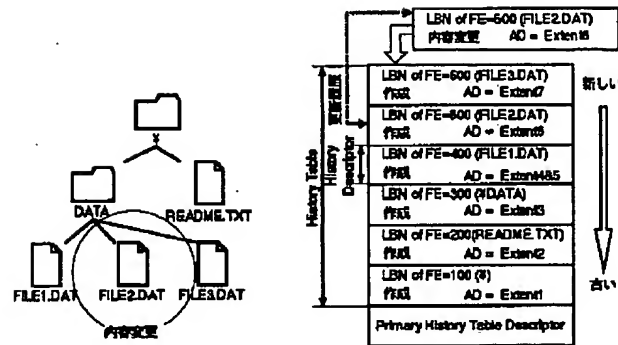
【図 22】



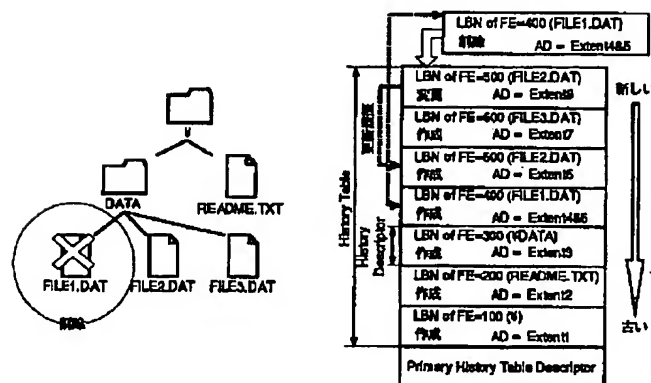
【図24】



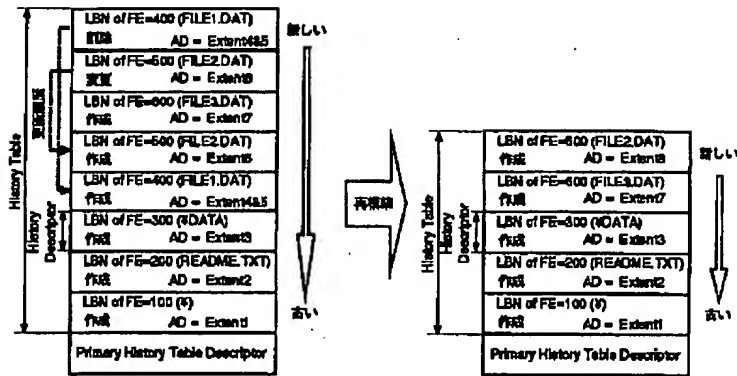
【図25】



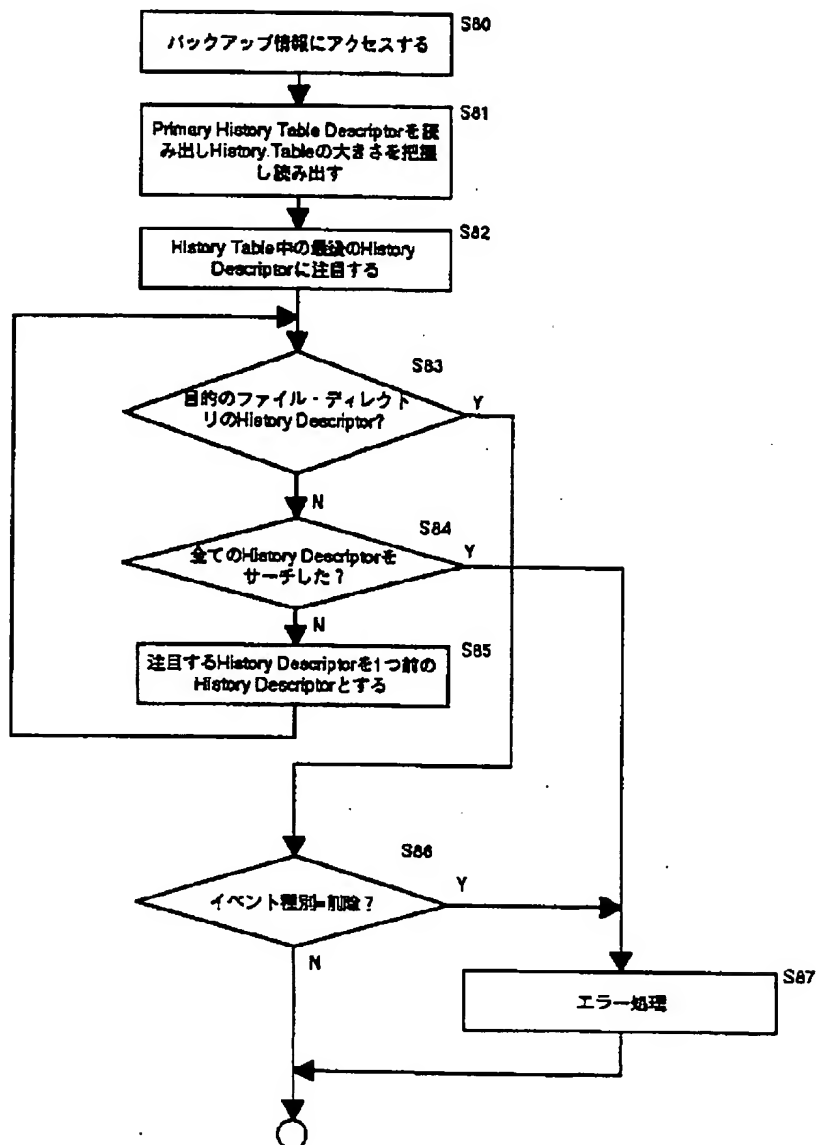
【図26】



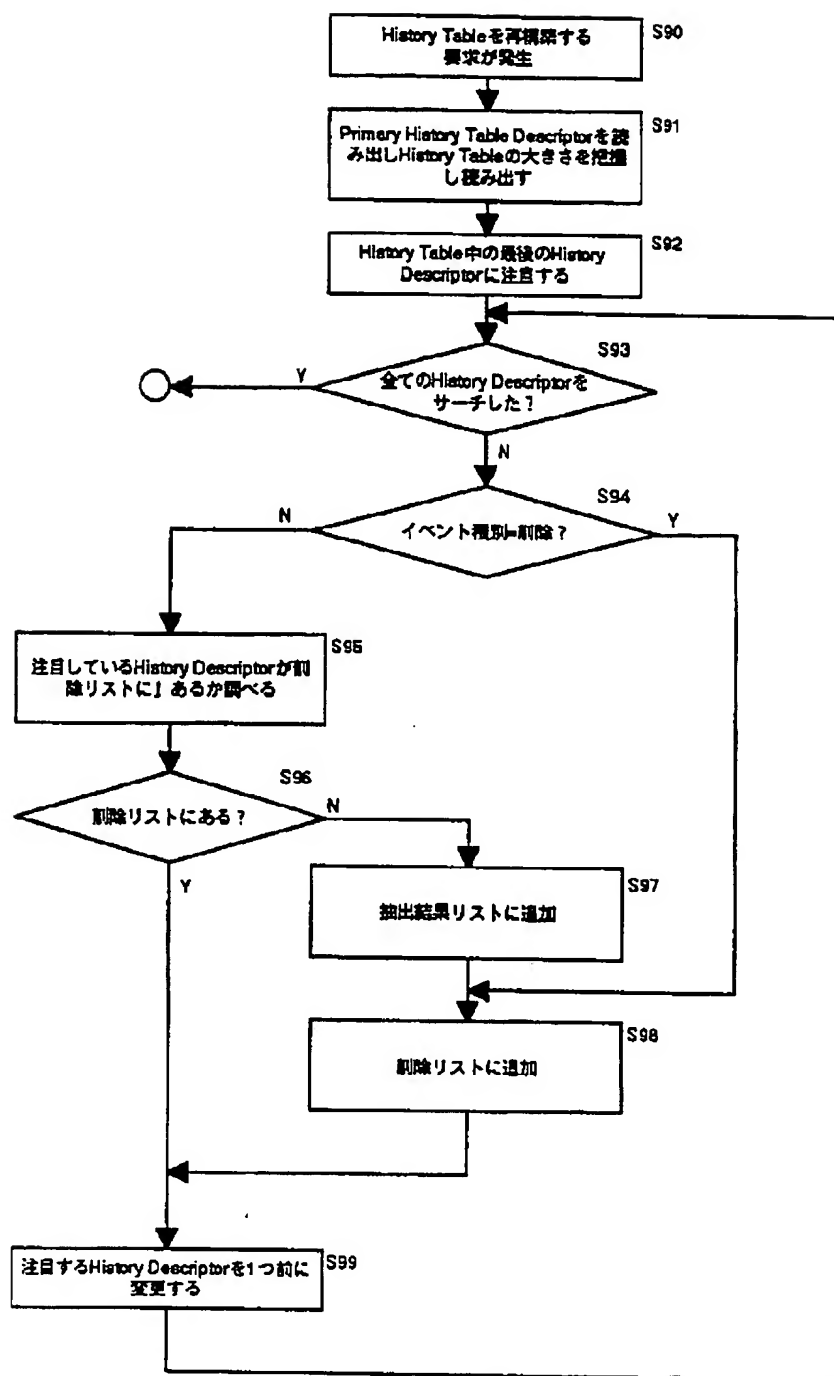
【図27】



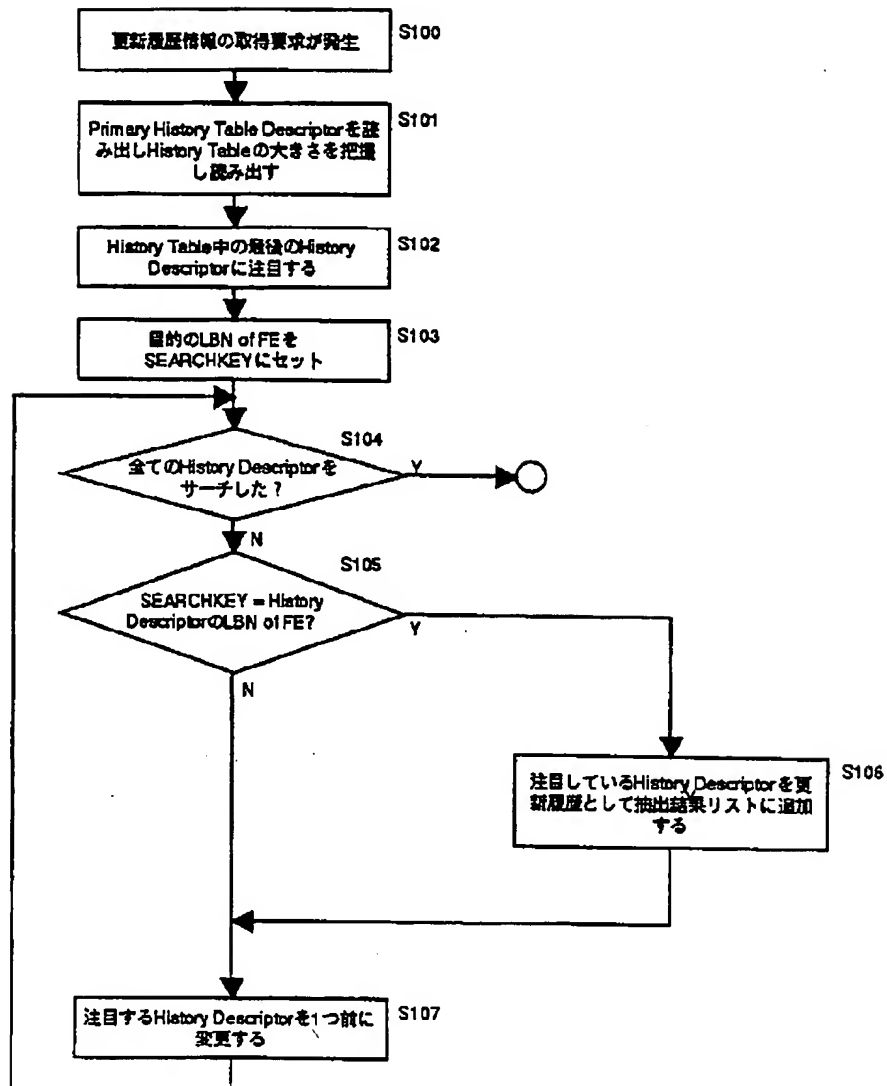
【図29】



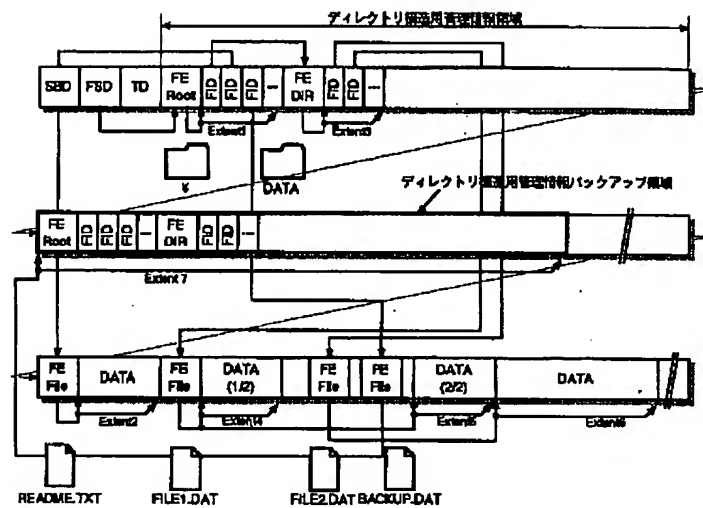
【図30】



【図31】



【図 33】



フロントページの続き

(72)発明者	西村 元秀	
	大阪府大阪市阿倍野区長池町22番22号	シ
	ャーブ株式会社内	
(72)発明者	木山 次郎	
	大阪府大阪市阿倍野区長池町22番22号	シ
	ャーブ株式会社内	
(72)発明者	山村 博幸	
	大阪府大阪市阿倍野区長池町22番22号	シ
	ャーブ株式会社内	

(72)発明者 山口 孝好
大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

(72)発明者 木村 英士
大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

F ターム(参考) 5B018 GA04 HA22 MA12
5B082 DE01 EA01 GA14

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.